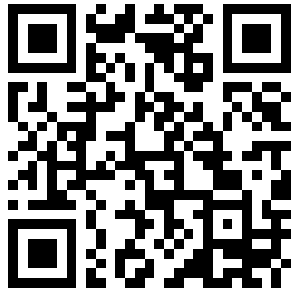


Principles of Knowledge Representation and Reasoning:

Proceedings of the Fifth International Conference (KR '96)

The Morgan Kaufmann Series in Representation and Reasoning

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

Books

James Allen, James Hendler, and Austin Tate, editors
Readings in Planning (1990)

James F. Allen, Henry A. Kautz, Richard N. Pelavin, and Josh D. Tenenberg
Reasoning About Plans (1991)

Ronald J. Brachman and Hector Levesque, editors
Readings in Knowledge Representation (1985)

Ernest Davis
Representations of Commonsense Knowledge (1990)

Thomas L. Dean and Michael P. Wellman
Planning and Control (1991)

Janet Kolodner
Case-Based Reasoning (1993)

Judea Pearl
Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference (1988)

Glenn Shafer and Judea Pearl, editors
Readings in Uncertain Reasoning (1990)

John Sowa, editor
Principles of Semantic Networks: Explorations in the Representation of Knowledge (1991)

Daniel S. Weld and Johan de Kleer, editors
Readings in Qualitative Reasoning about Physical Systems (1990)

David E. Wilkins
Practical Planning: Extending the Classical AI Paradigm (1988)

Proceedings

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

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

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

Principles of Knowledge Representation & Reasoning: Proceedings of the Fourth International Conference (KR '94)
Edited by Jon Doyle, Erik Sandewall, and Pietro Torasso

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

The Frame Problem in Artificial Intelligence: Proceedings of the 1987 Conference
Edited by Frank M. Brown (1987)

Reasoning About Actions and Plans: Proceedings of the 1986 Workshop
Edited by Michael P. Georgeff and Amy L. Lansky (1987)

Theoretical Aspects of Reasoning and Knowledge: Proceedings of the First Conference (TARK 1986)
Edited by Joseph P. Halpern

Theoretical Aspects of Reasoning and Knowledge: Proceedings of the Second Conference (TARK 1988)
Edited by Moshe Y. Vardi

Theoretical Aspects of Reasoning and Knowledge: Proceedings of the Third Conference (TARK 1990)
Edited by Rohit Parikh

Theoretical Aspects of Reasoning and Knowledge: Proceedings of the Fourth Conference (TARK 1992)
Edited by Yoram Moses

Theoretical Aspects of Reasoning and Knowledge: Proceedings of the Fifth Conference (TARK 1994)
Edited by Ronald Fagin

Theoretical Aspects of Rationality and Knowledge: Proceedings of the Sixth Conference (TARK 1996)
Edited by Yoav Shoam

Principles of Knowledge Representation and Reasoning:

Proceedings of the Fifth International Conference (KR '96)

Edited by

Luigia Carlucci Aiello
(Università di Roma "La Sapienza")

Jon Doyle
(Massachusetts Institute of Technology)

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

Morgan Kaufmann Publishers, Inc.
San Francisco, California

These proceedings were managed and produced for the organizers
of the KR '96 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 1-55860-421-9
ISSN 1046:9567

94 95 96 97 4 3 2 1

Contents

CONSTRAINTS

ROBOTICS

SPATIAL REPRESENTATION AND REASONING

RECOGNITION AND DIAGNOSIS

INHERITANCE

DESCRIPTION LOGICS: REASONING TECHNIQUES

DESCRIPTION LOGICS: EXPRESSIVITY AND COMPLEXITY

COMPLEXITY MEASURES

DEDUCTIVE SYSTEMS

BELIEF REVISION

PREFERENCE LOGIC

NONMONOTONIC REASONING

NONMONOTONIC LOGICS AND LOGIC PROGRAMMING

DECISION THEORY

REPORTS ON IMPLEMENTATIONS

PSYCHOLOGICAL AND PHILOSOPHICAL CONNECTIONS

INVITED TALKS

PANELS

Preface

We base ourselves on the idea that in order for a program to be capable of learning something it must first be capable of being told it.

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, 1958

This volume contains the papers presented at the Fifth 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.

KR '96 continues the tradition of high standards for accepted papers established by the preceding four conferences. We received 187 extended abstracts from 24 countries and 6 of the 7 continents (missing only Antartica). The 55 papers we were able to include in the conference come from 13 countries and 5 continents.

Many areas traditionally attracting strong KR interest, including description logics, nonmonotonic reasoning, action and time, remain well represented in KR '96, with the newer topic of spatial reasoning growing to join them. Continued interest in complexity is demonstrated by two sessions of submitted papers and the invited plenary talk by Georg Gottlob. Our attempt to expand the conference beyond the traditional extremely formal papers was rewarded this year by enough accepted papers to have two non-traditional sessions, one on "Reports on Implementations," which includes a report on a system that has been in use for 5 years at Ford Motor Company, and one on "Psychological and Philosophical Connections."

KR '96 marks the 40th anniversary of the 1956 Dartmouth Summer Research Project on Artificial Intelligence. It is therefore fitting that our opening invited plenary speaker is John McCarthy, who co-organized that meeting, who gave the field of AI its name, and whose seminal "Advice Taker" paper, from which are taken the opening quotes of this preface, exerted a huge and continuing influence on KR&R. It is also fitting that the situation calculus, introduced by McCarthy in a memorandum first distributed in 1963, and frequently reprinted together with the Advice Taker paper, has attracted so much renewed interest that there is a session in this conference devoted to it.

Luigia Carlucci Aiello
Program Co-Chair

Jon Doyle
Conference Chair

Stuart C. Shapiro
Program Co-Chair

Acknowledgments

KR '96 would not have been possible without the efforts of a great number of dedicated people.

First, and most important, is our outstanding program committee, who were asked to contribute extraordinary effort in reviewing and comparing 187 extended abstracts, and who did an excellent job:

Syed Ali
UWM, Wisconsin, USA

Fahiem Bacchus
U Waterloo, Canada

Afzal Ballim
EPFL, Switzerland

John A. Barnden
New Mexico SU, USA

Ronald J. Brachman
AT&T, USA

Maurice Bruynooghe
CU Leuven, Belgium

Anthony G. Cohn
U Leeds, UK

Marie Odile Cordier
IRISA, France

Ernest Davis
New York U, USA

Didier Dubois
IRIT, France

Thomas Eiter
TU Vienna, Austria

Luis Farinas del Cerro
IRIT, France

Richard Fikes
Stanford U, USA

Dov Gabbay
Imperial College, UK

Peter Gärdenfors
Lund U, Sweden

Michael Georgeff
AAII, Australia

Fausto Giunchiglia
U Trento, Italy

Patrick Hayes
U Illinois, USA

James Hendler
U Maryland, USA

Eduard Hovy
USC/ISI, USA

David Israel
SRI, USA

Lucja Iwanska
Wayne SU, USA

Hiroachi Kitano
Sony Labs, Japan

Kurt Konolige
SRI, USA

Sarit Kraus
Bar Ilan U, Israel

Benjamin Kuipers
U Texas, USA

Deepak Kumar
Bryn Mawr Coll, USA

Gerhard Lakemeyer
U Bonn, Germany

Fritz Lehmann
Cycorp & GRANDAI, USA

Doug Lenat
Cycorp, USA

Maurizio Lenzerini
U Roma, Italy

Hector Levesque
U Toronto, Canada

Vladimir Lifschitz
U Texas, USA

Robert MacGregor
USC/ISI, USA

João Martins
TU Lisbon, Portugal

Riichiro Mizoguchi
Osaka U, Japan

Bernard Nebel
U Frieburg, Germany

Hwee Tou Ng
DSO, Singapore

Hans J. Ohlbach
Imperial College, UK

Lin Padgham
RMIT U, Australia

Ramesh Patil
USC/ISI, USA

Anand Rao
AAII, Australia

Ray Reiter
U Toronto, Canada

Jeff Rosenschein
Hebrew U, Israel

Erik Sandewall
Linkoeping U, Sweden

Lenhart Schubert
U Rochester, USA

John Sowa
U Binghamton, USA

Piero Torasso
U Torino, Italy

Frank van Harmelen
Vrije U, The Netherlands

Wolfgang Wahlster
DFKI, Germany

The program committee was assisted by the following guest reviewers.

Mathias Bauer
Massimo Benerecetti
Philippe Besnard
Alexander Bochman
Paolo Bouquet
Marco Cadoli
Lawrence Cavedon
Hans Chalupsky
Alessandro Cimatti
Ido Dagan
Giuseppe De Giacomo
Marc Denecker
Uwe Egly
Luc de Raedt
Francesco Donini
Thomas Drakengren
Roman Englert
Enrico Franconi
Chiara Ghidini
Enrico Giunchiglia
Andrea Hemprich
Catholijn Jonker
Daniel Lehmann
Michael Leuschel
Andrei Mantsivoda
Susan McRoy
Leora Morgenstern
Kenneth Murray
Daniele Nardi
Jacques Nicolas
Fiora Pirri
Dominique Py
Jochen Renz
Jacques Riche
Andrea Schaerf
Marco Schaerf
Roberto Sebastiani
Pascale Sebillot
Luciano Serafini
Wolfgang Slany
Markus Stumptner
Annette ten Teije
Kristof Van Belleghem
Henk Vandecasteele
Toby Walsh

We thank the rest of the KR '96 Organizing Committee: Ron Loui, Inter-Conference Cooperation Chair; and Werner Horn, Publicity Chair.

We appreciate the willingness of John McCarthy and Georg Gottlob to be our invited speakers and the efforts they expended to set the tone of the conference. We also thank Richard Fikes, Ron Loui, and Bob MacGregor for organizing the three panel sessions.

We are indebted to Carol Hamilton, Julia Bowen, and Hollan Young of the American Association for Artificial Intelligence (AAAI), and to Jennifer Ballentine of Professional Book Center for keeping us on schedule, managing the flow of materials, and producing the printed materials, including these proceedings.

Situation Calculus

Natural Actions, Concurrency and Continuous Time in the Situation Calculus

Ray Reiter
Department of Computer Science
University of Toronto
Toronto, Canada M5S 1A4
and
The Canadian Institute for Advanced Research
email: reiter@cs.toronto.edu http://www.cs.toronto.edu/~cogrobo/

Abstract

Our focus in this paper is on natural exogenous actions (Pinto [23]), namely those which occur in response to known laws of physics, like a ball bouncing at times determined by Newtonian equations of motion. The property of such actions that we wish to capture is that they must occur at their predicted times, provided no earlier actions (natural or agent initiated) prevent them from occurring. Because several such actions may occur simultaneously, we need a theory of concurrency. Because such actions may be modeled by equations of motion, we need to represent continuous time. This paper shows how to gracefully accommodate all these features within the situation calculus, without sacrificing the simple solution to the frame problem of Reiter [25]. One nice consequence of this approach is a situation calculus specification of deductive planning, with continuous time and true concurrency, and where the agent can incorporate external natural event occurrences into her plans.

1 Introduction

For the past several years, the Cognitive Robotics Group at the University of Toronto has been exploring the feasibility of the situation calculus (McCarthy [19]) as a theoretical and computational foundation for modeling autonomous agents dwelling in dynamic environments. It is a challenging research problem to capture, in a single formal and computational framework, the full range of characteristics associated with such settings: the frame, ramification and qualification problems, exogenous and natural events, chance events and the unpredictability of action effects, complex actions and procedures and the ability of an agent to perform such actions, time, concurrency, hypothetical and counterfactual reasoning about action occurrences and time, perceptual actions and their effects on an agent's mental state, the complex relationships among reasoning, perception and action, planning, belief revision in the presence of conflicting observations, etc. The principal objective of this project is to provide just such a general theory of actions and time, and, as noted above, our formal foundation for this has been the situation calculus.

While we remain far from achieving these long-range objectives, we have had some modest success in this undertaking. Starting with a solution to the frame problem for deterministic, simple actions (Reiter [25]), we have defined and implemented a novel situation calculus-based logic programming language for defining complex agent behaviors (Levesque et al. [14]), and experimented with it in a robotics application (Lespérance et al. [12]), and for software agents (Marcu et al. [18]). Scherl and Levesque [28] have given a situation calculus account of sensing (knowledge-producing) actions, and Bacchus, Halpern and Levesque have extended this to include noisy sensors [2]. Lin [15] has extended Reiter's treatment for deterministic primitive actions to nondeterministic ones. Levesque [13] has given a situation calculus account of planning for agents which can sense their environments. Shapiro, Lespérance and Levesque [31] have formalized agent goals and rational actions in the situation calculus. Pinto [23] has proposed a situation calculus-based account of concurrency, natural actions and continuous time.

These initial results have encouraged us in our belief that the situation calculus is well suited to the general problem of providing a formal and computational account of complex dynamic domains and agent behaviors. This paper is a further step in the direction of appropriately enriching the situation calculus for this purpose. Specifically, we suitably modify, and build on, the earlier work mentioned above by Pinto [23] and also by Ternovskaia [32]. This we do by providing an axiomatization of the situation calculus to include concurrency, continuous time and two kinds of actions: (1) Those under the control of an autonomous agent

with the "free will" to perform any of these actions at any time, provided their preconditions are met. (2) Natural actions – those under nature's control – which must occur at their predetermined times provided no other actions (natural or agent initiated) occur earlier to prevent them from occurring. Towards that end, we begin by proposing an axiomatization of the concurrent, temporal situation calculus. With these axioms in hand, we define the *legal* situations, which are those that respect the property of natural actions that they must occur at their predicted times, unless something happens to prevent them. We then prove some intuitively desirable properties of these legal situations, for example, that worlds which lack agents with "free will" evolve deterministically, and we establish regression-based methods for verifying that a given situation is legal (the projection problem). One nice consequence of this approach is a situation calculus theory of deductive planning, with continuous time and true concurrency, and where the agent can incorporate external natural event occurrences into her plans.

2 Formal Preliminaries

2.1 The Language of the Situation Calculus

We begin by expanding the situation calculus ontology beyond that of Lin and Reiter [16] and Reiter [26]. The situation calculus is a sorted second order language with the following sorts, function and predicate symbols:

1. There is a sort *action* of *simple* actions. Conceptually, all simple actions are instantaneous, and every function symbol of sort *action* takes a parameter (in the last argument position) denoting the time of the action's occurrence. So, $start\_meeting(person, t)$ might be the instantaneous action of *person* starting a meeting at time t. This will make the notion of concurrent actions relatively unproblematic, which is not the case when actions have durations, and therefore may overlap in complicated ways (Gelfond, Lifschitz and Rabinov [7]).
2. A sort *time* ranging over the reals.
3. A function symbol *time*: $time(a)$ denotes the occurrence time of the simple action a.
4. A function symbol *start*: $start(s)$ denotes the start time of the situation s.
5. A new sort *concurrent*; these are sets of simple actions. We do not axiomatize sets, but instead rely on the standard interpretation of sets and their operations (union, intersection, etc) and relations (membership, subset, etc). This is in the same spirit as our treatment of the sort *time*; we do not axiomatize the reals for this purpose, but instead rely on the standard interpretation of the reals and their operations (addition, multiplication etc) and relations ($<$, $\leq$, etc). To distinguish the sorts *action* and *concurrent*, we use variables $a, a', \ldots$ and $c, c', \ldots$ respectively.
6. As in the sequential situation calculus, we have a sort *situation*, but now ranging over sequences of concurrent actions.
7. A binary function symbol do : *concurrent* $\times$ *situation* $\rightarrow$ *situation*, and a constant S_0 denoting the initial situation.
8. The sequential situation calculus has a distinguished predicate symbol $Poss$; $Poss(a, s)$ means that simple action a can be executed in the situation s. We extend $Poss$ to concurrent actions, and will write $Poss(c, s)$ to mean that the concurrent action c is possible in situation s.
9. The sequential situation calculus has a distinguished predicate symbol $<$. $s < s'$ means that one can get from situation s to situation s' by a sequence of executable (possible) simple actions. We extend $<$ to concurrent actions, so that $s < s'$ will mean that one can get from situation s to situation s' by a sequence of executable (possible) concurrent actions.
10. Finally, there are predicate symbols *natural*, *coherent*, *legal* and *lntp*, to be described later.

2.2 Foundational Axioms for the Concurrent, Temporal Situation Calculus

Lin and Reiter [16] and Reiter [26] provide foundational axioms for the sequential situation calculus. These need to be generalized to the concurrent, temporal setting, which we now do. Our assumption that all simple actions are instantaneous makes this generalization relatively unproblematic.

We begin by postulating a second order induction axiom:

$$(\forall P).P(S_0) \wedge (\forall c, s)[P(s) \supset P(do(c, s))] \supset (\forall s)P(s). \quad (1)$$

We need the following unique names axioms for situations:

$$S_0 \neq do(c, s),^1 \quad (2)$$

$$do(c, s) = do(c', s') \supset c = c' \wedge s = s'. \quad (3)$$

The time of an action occurrence is the value of that action's temporal argument. So, for each action function $A(\vec{x}, t)$ of our situation calculus language, we need an axiom:

$$time(A(\vec{x}, t)) = t, \quad (4)$$

[1] In what follows, lower case Roman characters will denote variables in formulas, unless otherwise noted. In addition, free variables will always be implicitly universally prenex quantified.

as, for example, in

$$time(start\_meeting(person, t)) = t.$$

Following Lin and Shoham [17], Pinto [23] and others we treat concurrent actions as sets, possibly infinite, of simple actions. As we'll see later, the possibility of infinitely many actions occurring concurrently must be taken seriously, so that the obvious notation $a_1 \| a_2 \| \cdots \| a_n$ cannot accommodate this possibility. Because concurrent actions are sets of simple actions, we can use the notation $a \in c$ to mean that simple action a is one of the actions of the concurrent action c.

We require that concurrent actions be coherent, which is to say, there is at least one action in the collection, and that all of the (instantaneous) actions in the collection occur at the same time:

$$coherent(c) \equiv (\exists a) a \in c \wedge (\exists t)(\forall a')[a' \in c \supset time(a') = t]. \quad (5)$$

We can now extend the function *time* from simple actions to concurrent ones, and we can define the function *start*, as follows:

$$coherent(c) \supset [time(c) = t \equiv (\exists a)(a \in c \wedge time(a) = t)] \wedge start(do(c, s)) = time(c). \quad (6)$$

Notice that we do not define the start time of S_0; this is arbitrary, and may (or may not) be specified to be any real number, depending on the application.

Not every action is executable in every situation. Accordingly, we introduce a binary predicate $Poss(c, s)$, meaning that it is possible to execute concurrent action c in situation s. What can we say in general about the preconditions of concurrent actions? At the very least, we need the following:

$$Poss(a, s) \supset Poss(\{a\}, s), \quad (7)$$

$$Poss(c, s) \supset coherent(c) \wedge (\forall a)[a \in c \supset Poss(a, s)]. \quad (8)$$

As we shall see in Section 3.3, the converse of (8) need not hold.

Finally, we need to reconsider the relation $<$ on situations as axiomatized for the sequential, non-temporal situation calculus in Lin and Reiter [16] and Reiter [26]. The intended interpretation of $s < s'$ is that situation s' is reachable from situation s by some sequence of one or more concurrent actions, each concurrent action of which is possible in that situation resulting from executing the actions preceding it in the sequence. Consider the situation

$$do(\{collide(B_1, B_2, 4), end\_lunch(Bill, 4)\}, do(\{start\_meeting(Susan, 6)\}, S_0)),$$

in which the time of the second action precedes that of the first. Intuitively, we do not want to consider such an action sequence possible, and we amend the foundational axioms for $<$ in the sequential, non-temporal case accordingly:

$$\neg s < S_0, \quad (9)$$

$$s < do(c, s') \equiv Poss(c, s') \wedge s \leq s' \wedge start(s') \leq time(c). \quad (10)$$

Here, $s \leq s'$ is an abbreviation for $s < s' \vee s = s'$. Now, $s < s'$ means that one can get to s' from s by a sequence of possible concurrent actions, and moreover, the times of those action occurrences must be nondecreasing. Notice that we are overloading the predicate $<$; it is used to order situations as well as real numbers in the temporal domain. It will always be clear from context which usage we mean. Finally, notice that the constraint $start(s') \leq time(c)$ in axiom (10) permits action sequences in which the time of an action may be the same as the time of a preceding action, without requiring that these actions occur concurrently. For example,

$$do(\{collide(B_1, B_2, 4), end\_lunch(Bill, 4)\}, do(\{start\_meeting(Susan, 4)\}, S_0)),$$

might be a perfectly good situation accessible from S_0. This situation is defined by a sequence of two concurrent actions, each of which has the same occurrence time. We allow for this possibility because 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 $start\_falling(t)$. Both actions occur at the same time, but conceptually, the falling event happens "after" the cutting. Accordingly, we want to allow the situation $do(\{start\_falling(t)\}, do(\{cut\_string(t)\}, S_0))$.

The axioms (1) - (10) are the foundational axioms for *the concurrent, temporal situation calculus.*

3 Axiomatizing Concurrent Worlds

Most actions (picking up a block, going from one location to another) take time. What use, then, is a theory of actions in which all actions are instantaneous? As observed by Pinto [23] and Ternovskaia [32], the trick for making this work in the situation calculus is to conceive of such actions as *processes*, represented by fluents, and to introduce durationless actions which initiate and terminate these processes. For example, in a blocks world, we might have instantaneous actions $start\_pickup(x, t)$ and $end\_pickup(x, t)$, and the process of picking up x is represented by the fluent $picking\_up(x, t, s)$. $start\_pickup(x, t)$ causes the fluent $picking\_up$ to be true, $end\_pickup(x, t)$ causes it to be false. In those situations s in which $picking\_up(x, t, s)$ is true, we can describe those properties of the world, for example the position of the agent's hand as a function of t, which must be true during the evolution of the process $picking\_up$.

3.1 Successor State Axioms

Reiter [25], building on the ideas of Pednault [21] and of Haas [9] and Schubert [29], proposes a solution to the frame problem for deterministic, nonconcurrent actions in the absence of state constraints. This provides a systematic way of obtaining so-called *successor state axioms* from the effect axioms. We have to generalize these successor state axioms slightly, to take concurrency into account. This is quite straightforward, following the proposals of Pinto in his Ph.D. thesis [23] and Ternovskaia [32]. So, we will write formulas like:

$$\begin{aligned} Poss(c,s) \supset [&picking\_up(x, do(c,s)) \equiv \\ &(\exists t) start\_pickup(x,t) \in c \vee \\ &picking\_up(x,s) \wedge \neg(\exists t) end\_pickup(x,t) \in c]. \end{aligned}$$

A more interesting example is due to James Allen [1]. Imagine a door with a spring latch. The door can be unlocked by turning the latch, but the agent must keep the latch turned, for if not, the spring loaded mechanism returns the latch to its locked position. To open the door, the agent must turn the latch, and keep it turned while she pushes on the door. The concurrent latch turning and door pushing causes the door to open. Neither action by itself will open the door. This is easy to do in the situation calculus if we view the action of turning and holding the latch open, which intuitively would have a duration, as a composite of two instantaneous actions, $turn\_latch(t)$ and $release\_latch(t)$, whose effects are to make the fluent $locked(s)$ false and true respectively. In the same spirit, we treat the action of pushing on a door, which also would intuitively have a duration, as a composite of two instantaneous actions $start\_push(t)$ and $end\_push(t)$, whose effects are to make the fluent $pushing(s)$ true and false respectively. The appropriate successor state axiom for *open* is:

$$\begin{aligned} Poss(c,s) \supset [&open(do(c,s)) \equiv \\ &(\exists t)[turn\_latch(t) \in c \wedge start\_push(t) \in c] \vee \\ &pushing(s) \wedge (\exists t) turn\_latch(t) \in c \vee \\ &\neg locked(s) \wedge (\exists t) start\_push(t) \in c \vee open(s)]. \end{aligned}$$

Those for *pushing* and *locked* are:

$$\begin{aligned} Poss(c,s) \supset [&pushing(do(c,s)) \equiv (\exists t) start\_push(t) \in c \\ &\vee pushing(s) \wedge \neg(\exists t) end\_push(t) \in c], \end{aligned}$$

$$\begin{aligned} Poss(c,s) \supset [&locked(do(c,s)) \equiv (\exists t) release\_latch(t) \in c \\ &\vee locked(s) \wedge \neg(\exists t) turn\_latch(t) \in c]. \end{aligned}$$

Another interesting example is due to Rob Miller.[2] Turning on the hot water faucet causes hot water to run (denoted by the fluent $hot(s)$); similarly for turning on the cold. Both the hot and cold water faucets share a common spout, so if only the hot water is running, you will burn your hand. This example is of interest because the occurrence of the action of turning on the cold water faucet *cancels* the effect of a co-occurrence of turning on the hot.

$$\begin{aligned} Poss(c,s) \supset [&hot(do(c,s)) \equiv (\exists t) turnonHot(t) \in c \vee \\ &hot(s) \wedge \neg(\exists t) turnoffHot(t) \in c]. \end{aligned}$$

$$\begin{aligned} Poss(c,s) \supset [&cold(do(c,s)) \equiv (\exists t) turnonCold(t) \in c \vee \\ &cold(s) \wedge \neg(\exists t) turnoffCold(t) \in c]. \end{aligned}$$

The following successor state axiom captures the conditions for burning oneself:

$$\begin{aligned} Poss(c,s) \supset [&burn(do(c,s)) \equiv \\ &\neg cold(s) \wedge (\exists t) turnonHot(t) \in c \wedge \\ &\quad \neg(\exists t) turnonCold(t) \in c \\ &\vee \neg hot(s) \wedge cold(s) \wedge (\exists t) turnonHot(t) \in c \wedge \\ &\quad (\exists t) turnoffCold(t) \in c \\ &\vee hot(s) \wedge cold(s) \wedge \neg(\exists t) turnoffHot(t) \in c \wedge \\ &\quad (\exists t) turnoffCold(t) \in c \\ &\vee burn(s)]. \end{aligned}$$

3.2 Action Precondition Axioms

The approach of Reiter [25] to axiomatizing dynamic worlds in the situation calculus relies on a collection of *action precondition axioms*, one for each action type, and we also rely on such axioms here. These specify necessary and sufficient conditions under which the action is possible. For example, in a blocks world, we might have the following action precondition axiom for the action $pickup(x)$, by a one-handed robot:

$$\begin{aligned} Poss(pickup(x), s) \equiv [&(\forall y) \neg holding(y,s)] \wedge \\ &clear(x,s) \wedge \neg heavy(x,s). \end{aligned}$$

In general, an action precondition axiom will have the syntactic form:

$$Poss(A(\vec{x},t),s) \equiv \Phi(\vec{x},t,s). \qquad (11)$$

Here, $\Phi(\vec{x},t,s)$ is any first order formula with free variables among $\vec{x}, t$ and s whose only term of sort *situation* is s.

3.3 The Precondition Interaction Problem

Our approach to axiomatizing actions appeals to *action precondition axioms* for specifying the preconditions of simple actions. These have the syntactic form (11). As pointed out by Pelavin [22] and Pinto [23], in the case of action preconditions for concurrent actions, the converse of (8) need not hold. Two simple actions may each be possible, their action preconditions may be jointly consistent, yet intuitively they should not be concurrently possible. Pinto calls this the *precondition interaction problem*. Here is a simple example, after a similar example of Pelavin [22]:

$$Poss(start\_move\_left(t), s) \equiv \neg moving(s).$$

$$Poss(start\_move\_right(t), s) \equiv \neg moving(s).$$

[2] Personal communication.

Intuitively,
$Poss(\{start\_move\_left(t), start\_move\_right(t)\}, s)$
should be false. With reasonable successor state axioms, we should be able to derive something like:

$$Poss(\{start\_move\_left(t), start\_move\_right(t)\}, s) \supset moving\_right(do(\{start\_move\_left(t), start\_move\_right(t)\}, s)),$$

and

$$Poss(\{start\_move\_left(t), start\_move\_right(t)\}, s) \supset moving\_left(do(\{start\_move\_left(t), start\_move\_right(t)\}, s)).$$

So, in the presence of a reasonable state constraint like:

$$\neg[moving\_right(s) \wedge moving\_left(s)],$$

we could derive

$$\neg Poss(\{start\_move\_left(t), start\_move\_right(t)\}, s).$$

These are complicated issues which need to be looked at more closely; see Pinto [23] for a more extensive discussion. We shall ignore them here, except to note that the ideas in the rest of this paper require no commitment, one way or another, to the form of a solution to the precondition interaction problem.

3.4 Infinitely Many Actions Can Co-Occur

Nothing prevents one from writing:

$$Poss(A(x,t), s) \equiv t = 1,$$

in which case $A(x, 1)$ can co-occur, for all x. So if x ranges over the natural numbers (or the reals, or ...) we get lots of possible co-occurrences.

4 Natural Actions

Our focus in this paper is on natural exogenous actions (Pinto [23]), namely those which occur in response to known laws of physics, like a ball bouncing at times determined by Newtonian equations of motion. These laws of physics will be embodied in the action precondition axioms, in the style of Pinto's PhD thesis [23], but in a somewhat more natural form:

$$Poss(bounce(t), s) \equiv is\_falling(s) \wedge \{height(s) + vel(s)[t - start(s)] - 1/2g[t - start(s)]^2 = 0\}.$$

Here, $height(s)$ and $vel(s)$ are the height and velocity, respectively, of the ball at the start of situation s.

Notice that the truth of $Poss(bounce(t), s)$ *does not* mean that the bounce action must occur in situation s, or even that the bounce action must eventually occur. It simply means that the bounce is physically possible at time t in situation s; a *catch* action occurring before t should prevent the bounce action.

We introduce a predicate symbol *natural*, with which the axiomatizer can declare suitable actions to be natural, as, for example, $natural(bounce(t))$.

4.1 Natural Actions and Legal Situations

In the space of all possible situations, we want to single out the legal situations, i.e. those which respect the property of natural actions that they must occur at their predicted times, provided no earlier actions (natural or agent initiated) prevent them from occurring. We capture these legal situations with the following definition:

$$legal(s) \equiv [S_0 \leq s \wedge (\forall a, c, s').natural(a) \wedge Poss(a, s') \wedge do(c, s') \leq s \supset a \in c \vee time(c) < time(a)]. \quad (12)$$

This definition may initially be a bit difficult to understand; the following provides a more intuitive inductive characterization of the legal situations.

Lemma 1 *The foundational axioms imply that the definition (12) is equivalent to the conjunction of the following two sentences:*

$$legal(S_0).$$

$$legal(do(c, s)) \equiv [legal(s) \wedge Poss(c, s) \wedge start(s) \leq time(c) \wedge (\forall a).natural(a) \wedge Poss(a, s) \supset a \in c \vee time(c) < time(a)].$$

Proof:
$\Rightarrow$ Straightforward.
$\Leftarrow$ Use the induction axiom (1), with the definition (12) as induction hypothesis.

4.2 An Example: Enabling Actions

In the discussion following the presentation of axiom (10), we noted the possibility of situations containing two or more concurrent actions with the same occurrence times. We now provide an example where this is a desirable feature of our axiomatization. Consider a scenario in which an agent is holding an object. At some time she releases the object, enabling it to start falling. The $start\_falling$ action is a natural action, which is to say, it must occur immediately after the release action. For simplicity, assume that once the object starts to fall, it continues falling forever.

$$start(S_0) = 0, \quad holding(S_0), \quad \neg falling(S_0).$$
$$natural(a) \equiv (\exists t)a = start\_falling(t),$$
$$Poss(release(t), s) \equiv holding(s) \wedge start(s) \leq t,$$
$$Poss(start\_falling(t), s) \equiv \neg holding(s) \wedge \neg falling(s) \wedge start(s) \leq t,$$
$$Poss(c, s) \supset [falling(do(c, s)) \equiv (\exists t)start\_falling(t) \in c \vee falling(s)],$$
$$Poss(c, s) \supset [holding(do(c, s)) \equiv (\exists t)catch(t) \in c \vee holding(s) \wedge \neg(\exists t)release(t) \in c].$$

Then, the following is a legal situation:

$$do(\{start\_falling(1)\}, do(\{release(1)\}, S_0)).$$

The following is *not* a legal situation:

$$do(\{start\_falling(2)\}, do(\{release(1)\}, S_0)).$$

4.3 Zeno's Paradox

Legal situations admit infinitely many distinct action occurrences over a finite time interval. Consider the natural action A:

$$Poss(A(t), s) \equiv t = (1 + start(s))/2,$$

with $start(S_0) = 0$. Then for any $n \geq 1$, the situation $do([A(1/2), \ldots, A(1 - 1/2^n)], S_0)$ is legal.[3] This means that if B is another action, natural or not, with $Poss(B(t), s) \equiv t = 1$, then $B(1)$ never gets to be part of any legal situation; it never happens! This is arguably the right intuition, given the idealization of physical reality involved in the axiomatization of A. There does not appear to be any simple way to prevent Zeno's paradox from arising in temporal axiomatizations like ours. Of course, this is not really a paradox, in the sense that such examples do not introduce any inconsistencies into the axiomatization. See E. Davis [4] for a deeper discussion of these issues.

4.4 The Natural World Condition

This is the sentence:

$$(\forall a) natural(a). \qquad (NWC)$$

The Natural World Condition restricts the domain of discourse to natural actions only.

Lemma 2 *The following is a consequence of the foundational axioms and the definition (12):*

$$legal(do(c, s)) \wedge legal(do(c', s)) \wedge NWC \supset c = c'.$$

Proof:
Suppose, for fixed c, c', s, that $legal(do(c, s))$, $legal(do(c', s))$, and NWC. Then, by Lemma 1, $Poss(c, s)$, and $Poss(c', s)$, and therefore, by (8), $coherent(c)$ and $coherent(c')$.

1. First we prove $a \in c \supset a \in c'$. Suppose, to the contrary, that there is some α, with $\alpha \in c$, but $\alpha \notin c'$. By NWC, $natural(\alpha)$. Since $Poss(c, s)$, we have $Poss(\alpha, \sigma)$ by (8). Hence, since $legal(do(c', s))$, we conclude, with the help of Lemma 1, that

   $$time(c') < time(\alpha). \qquad (13)$$

   Since $coherent(c')$, c' is nonempty by (5), so there exists β such that $\beta \in c'$. Since $Poss(c', s)$, by (8), we conclude $Poss(\beta, s)$. Since $natural(\beta)$, we conclude, by Lemma 1, that

   $$\beta \in c \vee time(c) < time(\beta). \qquad (14)$$

   But, since $coherent(c)$ and $coherent(c')$, we have $time(\alpha) = time(c)$ and $time(\beta) = time(c')$, and these, together with (13), imply $time(\beta) < time(c)$. This, together with (14), implies $\beta \in c$. Since $coherent(c)$, $time(\beta) = time(c)$, which contradicts $time(\beta) < time(c)$.
2. The proof that $a \in c' \supset a \in c$ is entirely symmetric to the previous one.

Combining 1 and 2, we conclude $c = c'$.

□

Intuitively, the above lemma tells us that natural worlds are *deterministic*: If there is a legal successor situation, it is unique. The following theorem extends Lemma 2 to *histories*: When there are only natural actions, the world evolves in a unique way, if it evolves at all.

Theorem 1 *The foundational axioms and the definition (12) entail the following:*

$$legal(s) \wedge legal(s') \wedge NWC \supset S_0 \leq s \leq s' \vee S_0 \leq s' \leq s.$$

Proof:
The proof is by induction on s, using the induction axiom (1). The case $s = S_0$ is immediate, so assume the result for s, and suppose $legal(do(c, s)) \wedge legal(s') \wedge NWC$. We must prove $S_0 \leq do(c, s) \leq s' \vee S_0 \leq s' \leq do(c, s)$. Since $legal(do(c, s))$, then by Lemma 1, $legal(s)$ and $Poss(c, s)$. Hence, by the induction hypothesis, we conclude $S_0 \leq s \leq s' \vee S_0 \leq s' \leq s$.
Case 1: $S_0 \leq s \leq s'$.
Case 1.1: $s = s'$. Then, because $Poss(c, s)$, $s' < do(c, s)$ by axiom (10), so, $S_0 \leq s' \leq do(c, s)$.
Case 1.2: $s < s'$.
We require the following two results, each of which is provable by induction on s'.

$$(\forall s, s').s < s' \supset (\exists c) do(c, s) \leq s', \qquad (15)$$

$$(\forall s, s').legal(s') \wedge s \leq s' \supset legal(s). \qquad (16)$$

Now, by (15), $do(c', s) \leq s'$ for some c'. Moreover, because $legal(s')$, we have, by (16), $legal(do(c', s))$. Since also $legal(do(c, s))$, then by Lemma 2, $c = c'$, and we conclude that $S_0 \leq do(c, s) \leq s'$.
Case 2: $S_0 \leq s' \leq s$.
Since $Poss(c, s)$, then by axiom (10), $s' < do(c, s)$, and by the transitivity of $\leq$ (provable by induction), we conclude $S_0 \leq s' \leq do(c, s)$.

4.5 Least Natural Time Points

The following definition plays a central role in theorizing about natural actions:

$$\begin{aligned} &lntp(s, t) \equiv \\ &(\exists a)[natural(a) \wedge Poss(a, s) \wedge time(a) = t] \wedge \\ &(\forall a)[natural(a) \wedge Poss(a, s) \supset time(a) \geq t]. \end{aligned} \qquad (17)$$

[3] $do([a_1, \ldots, a_n], s)$ abbreviates the situation reached from s by performing the actions $a_1, \ldots, a_n$ in sequence.

Intuitively, the *least natural time point* is the earliest time during situation s at which a natural action can occur.

Remark 1 *(17) entails the following:*

$$lntp(s,t) \wedge lntp(s,t') \supset t = t'.$$

So, when it exists, the least natural time point is unique. The least natural time point need not exist, for example, when $(\forall a).natural(a) \equiv (\exists x,t)a = B(x,t)$, where x ranges over the nonzero natural numbers, and $Poss(B(x,t),s) \equiv t = start(s) + 1/x$.

Lemma 3 *Our situation calculus axioms entail:*
$natural(a) \wedge legal(do(c,s)) \wedge a \in c \supset lntp(s, time(a))$.

Proof:
Assume that $natural(a)$, $legal(do(c,s))$ and $a \in c$, for fixed a, c, s. Since $legal(do(c,s))$, we know, by Lemma 1, that $Poss(c,s)$, and therefore, by (8) that $Poss(a,s)$ and $coherent(c)$. We must prove $lntp(s, time(a))$. By the definition of $lntp$, it is sufficient to prove $(\forall a').natural(a') \wedge Poss(a',s) \supset time(a') \geq time(a)$. So, for fixed a', assume $natural(a') \wedge Poss(a',s)$. We prove $time(a') \geq time(a)$. Since $legal(do(c,s))$, we know, by Lemma 1, that

$$a' \in c \vee time(c) < time(a'). \tag{18}$$

Moreover, since $coherent(c)$ and since $a \in c$, we conclude from (5) that $time(a) = time(c)$. (18) gives two cases:
Case 1: $time(c) < time(a')$. Since $time(a) = time(c)$, we have $time(a') > time(a)$.
Case 2: $a' \in c$. Since $coherent(c)$, we infer $time(a') = time(c)$. Since $time(a) = time(c)$, then $time(a') = time(a)$.

□

So, whenever $do(c,s)$ is legal, c's natural actions all co-occur at the least natural time point of s. All the actions that must co-occur first, according to the "laws of motion", actually do co-occur.

In the case of a domain closure assumption on natural actions, we can give an explicit formula for $lntp(s,t)$. So, suppose we have the following domain closure axiom:

$$natural(a) \equiv (\exists \vec{x},t)a = A_1(\vec{x},t) \vee \cdots \vee (\exists \vec{z},t)a = A_n(\vec{z},t), \tag{19}$$

together with the associated declarations (4):

$$\begin{array}{c} time(A_1(\vec{x},t)) = t, \\ \vdots \\ time(A_n(\vec{z},t)) = t. \end{array} \tag{20}$$

Lemma 4 *(17), (19) and (20) entail the following:*

$$\begin{array}{l} lntp(s,t) \equiv \\ [(\exists \vec{x})Poss(A_1(\vec{x},t),s) \vee \cdots \vee (\exists \vec{z})Poss(A_n(\vec{z},t),s)] \wedge \\ (\forall \vec{x},t')[Poss(A_1(\vec{x},t'),s) \supset t' \geq t] \wedge \cdots \wedge \\ (\forall \vec{z},t')[Poss(A_n(\vec{z},t'),s) \supset t' \geq t]. \end{array}$$

The Least Natural Time Point Condition

In view of the possibility of "pathological" axiomatizations, for which the least natural time point may not exist (see comments following Remark 1), we introduce the following sentence:

$$(\forall s).(\exists a)[natural(a) \wedge Poss(a,s)] \supset (\exists t)lntp(s,t). \tag{LNTPC}$$

Normally, it will be the responsibility of the axiomatizer to prove, usually by induction, that his axioms entail $LNTPC$.

Theorem 2 *Our situation calculus axioms entail:*

$$\begin{array}{l} LNTPC \supset \\ legal(do(c,s)) \equiv \{legal(s) \wedge Poss(c,s) \wedge \\ \quad start(s) \leq time(c) \wedge \\ \quad [(\forall a).natural(a) \wedge time(a) \leq time(c) \supset \\ \quad\quad [a \in c \equiv Poss(a,s) \wedge lntp(s, time(a))]]\}. \end{array}$$

Proof:
⇒
Assume $LNTPC$, and for fixed c, s, $legal(do(c,s))$. By Lemma 1, we conclude $legal(s)$, $Poss(c,s)$ and $start(s) \leq time(c)$. So we must prove:

$$\begin{array}{l} (\forall a).natural(a) \wedge time(a) \leq time(c) \supset \\ \quad [a \in c \equiv Poss(a,s) \wedge lntp(s, time(a))]. \end{array}$$

So, for fixed a, assume $natural(a) \wedge time(a) \leq time(c)$; we prove: $a \in c \equiv Poss(a,s) \wedge lntp(s, time(a))$.

1. Assume $a \in c$; we prove $Poss(a,s) \wedge lntp(s, time(a))$. Since $Poss(c,s)$, we know $Poss(a,s)$ by (8). By Lemma 3, we know $lntp(s, time(a))$.
2. Assume $Poss(a,s) \wedge lntp(s, time(a))$; we prove $a \in c$. Since $legal(do(c,s))$, then by Lemma 1: $a \in c \vee time(c) < time(a)$. Since $time(a) \leq time(c)$, we conclude $a \in c$.

⇐
Assume $LNTPC$, and for fixed c, s, $legal(s)$, $Poss(c,s)$, $start(s) \leq time(c)$, and

$$\begin{array}{l} (\forall a).natural(a) \wedge time(a) \leq time(c) \supset \\ \quad [a \in c \equiv Poss(a,s) \wedge lntp(s, time(a))]. \end{array} \tag{21}$$

By Lemma 1, we must prove: $(\forall a').natural(a') \wedge Poss(a',s) \supset a' \in c \vee time(c) < time(a')$. So, for fixed a', assume $natural(a') \wedge Poss(a',s) \wedge time(a') \leq time(c)$. We must prove that $a' \in c$. By (21), it is sufficient to prove $Poss(a',s) \wedge lntp(s, time(a'))$. We already know that $Poss(a',s)$, so we must prove $lntp(s, time(a'))$. By $LNTPC$, we have $lntp(s,\tau)$ for some τ, so there is an α such that $natural(\alpha) \wedge Poss(\alpha,s) \wedge time(\alpha) = \tau$. We prove that $time(a') = time(\alpha)$, from which the desired conclusion $lntp(s, time(a'))$ follows. Now, because $Poss(a',s)$ and $natural(a')$, we have $time(a') \geq time(\alpha)$. Therefore, because $time(a') \leq time(c)$, $time(\alpha) \leq time(c)$,

so by (21), $\alpha \in c$. Since $Poss(c, s)$, by (8) and (5), $time(\alpha) = time(c)$. Since $time(a') \geq time(\alpha) = time(c)$, and since also $time(a') \leq time(c)$, we conclude $time(a') = time(\alpha)$. Hence, we have proved $a' \in c$.

Theorem 3 *Our situation calculus axioms entail the following:*

$$LNTPC \wedge NWC \supset$$
$$legal(do(c, s)) \equiv \{legal(s) \wedge Poss(c, s) \wedge$$
$$start(s) \leq time(c) \wedge$$
$$(\forall a)[a \in c \equiv Poss(a, s) \wedge lntp(s, time(a))]\}.$$

Proof:
The $\Leftarrow$ direction follows immediately from Theorem 2. To prove the $\Rightarrow$ direction, assume that $LNTPC$ and NWC, and, for fixed c and s, that $legal(do(c, s))$. We must prove: $(\forall a).a \in c \equiv Poss(a, s) \wedge lntp(s, time(a))$. By Lemma 1, we have $Poss(c, s)$, and

$$(\forall a).Poss(a, s) \supset a \in c \vee time(c) < time(a). \quad (22)$$

1. Suppose $a \in c$. Since $Poss(c, s)$, then by (8) we know that $Poss(a, s)$. It remains to prove $lntp(s, time(a))$. This follows immediately from Lemma 3.
2. Assume $Poss(a, s) \wedge lntp(s, time(a))$. We must prove $a \in c$. Since $Poss(c, s)$, then by (8), $coherent(c)$ and hence c is nonempty. So, there exists α such that $\alpha \in c$. Since $Poss(c, s)$, then by (8), we know that $Poss(\alpha, s)$ and $coherent(c)$, so by (5), $time(\alpha) = time(c)$. Since $lntp(s, time(a))$, we have that $time(\alpha) = time(c) \geq time(a)$. So, by (22), we conclude $a \in c$.

□

This theorem informs us that for natural worlds satisfying $LNTPC$, we obtain the next legal situation from the current one by assembling into c all the possible actions occurring at the least natural time point of the current situation, provided this collection of natural actions is possible, and the least natural time point is greater than or equal to the start time of the current situation. Intuitively, this is as it should be for natural worlds. Theorem 3 provides the theoretical foundation for a situation calculus-based simulator for physical systems (Kelley [10]).

The Concurrent Natural Actions Assumption

This is the following sentence:

$$coherent(c) \wedge (\forall a)[a \in c \supset natural(a) \wedge Poss(a, s)]$$
$$\supset Poss(c, s). \quad (CNAA)$$

This says that a coherent collection of natural actions is possible if each individual action in the collection is possible. In other words, the precondition interaction problem does not arise for co-occurring natural actions. This seems to be an assumption about the accuracy with which the physics of the world has been modeled by "equations of motion", in the sense that if these equations predict a co-occurrence, then this co-occurrence really happens in the physical world, so that in our situation calculus model of that world, this co-occurrence should be possible.

Using this and (8), we obtain the following:

Lemma 5 *Our situation calculus axioms entail:*

$$CNAA \wedge (\forall a)[a \in c \supset natural(a)] \supset$$
$$Poss(c, s) \equiv coherent(c) \wedge (\forall a)[a \in c \supset Poss(a, s)].$$

Corollary 1 *Our situation calculus axioms entail:*

$$LNTPC \wedge NWC \wedge CNAA \supset$$
$$legal(do(c, s)) \equiv \{legal(s) \wedge (\exists a)a \in c \wedge$$
$$start(s) \leq time(c) \wedge$$
$$(\forall a)[a \in c \equiv Poss(a, s) \wedge lntp(s, time(a))]\}.$$

Proof:
By Theorem 3, it is sufficient to prove that

$$LNTPC \wedge NWC \wedge CNAA \supset$$
$$(\exists a)a \in c \wedge start(s) \leq time(c) \wedge$$
$$(\forall a)[a \in c \equiv Poss(a, s) \wedge lntp(s, time(a))]$$
$$\equiv$$
$$Poss(c, s) \wedge start(s) \leq time(c) \wedge$$
$$(\forall a)[a \in c \equiv Poss(a, s) \wedge lntp(s, time(a))].$$

So, assume $LNTPC \wedge NWC \wedge CNAA$. The $\Leftarrow$ direction is straightforward, using Lemma 5, and the definition (5) of *coherent*. To prove the $\Rightarrow$ direction, assume, for fixed a, c, and s, that $(\exists a)a \in c$, and $(\forall a)[a \in c \equiv Poss(a, s) \wedge lntp(s, time(a))]$. It is sufficient to prove $Poss(c, s)$. By Lemma 5 and NWC, it is sufficient to prove $coherent(c) \wedge (\forall a)[a \in c \supset Poss(a, s)]$. The second conjunct follows from NWC, so we must prove $coherent(c)$, which is equivalent, by (5), to $(\exists a)a \in c \wedge (\exists t)(\forall a)[a \in c \supset time(a) = t]$. The first conjunct is given by assumption, so we are left with proving $(\exists t)(\forall a)[a \in c \supset time(a) = t]$. Suppose, for the purposes of deriving a contradiction, that $(\forall t)(\exists a).a \in c \wedge time(a) \neq t$. By assumption, $(\exists a)a \in c$, so, for fixed α, suppose $\alpha \in c$. Then we must have $(\exists a).a \in c \wedge time(a) \neq time(\alpha)$, so for some fixed α', we have $\alpha' \in c \wedge time(\alpha') \neq time(\alpha)$. Because $\alpha \in c$ and $\alpha' \in c$, we infer $lntp(s, time(\alpha))$ and $lntp(s, time(\alpha'))$, so by Remark 1, $time(\alpha') = time(\alpha)$, contradiction.

5 Some Consequences of this Approach

5.1 Planning with Concurrent and Natural Actions

The classical specification of the planning task is by Green [8], and concerns a single agent in complete control of all actions that can be performed in the world

being modeled. A ground situation term σ is a *plan* for G iff

$$Axioms \models S_0 \leq \sigma \wedge G(\sigma).$$

Here, *Axioms* provide the relevant background theory.

In view of the approach of this paper, we can now generalize Green's definition to the case of a single agent with the "free will" to perform a repertoire of actions under her control, and a complementary set of natural actions under nature's control: A ground situation term σ is a *plan* for G iff

$$Axioms \models legal(\sigma) \wedge G(\sigma).$$

Here, *Axioms* includes the foundational axioms and the associated definitions of this paper. It will also include action precondition and successor state axioms for the actions under consideration, unique names axioms for actions, and axioms specifying the initial world situation.

This means we now have a situation calculus specification of deductive planning, with continuous time and true concurrency, and where the agent can incorporate external natural event occurrences into her plans.

With the exception of Levesque's work on planning for agents with perceptual actions [13], this appears to be the first significant generalization of Green's classical formulation of deductive planning.

5.2 Regression

Lemma 1 provides a basis for establishing legality by regression (Waldinger [33], Pednault [21], Reiter [25]). When $LNTPC$ holds, Theorem 2 provides a better regression mechanism, and when also NWC is true, we can use Theorem 3.

While our focus in the previous section was on *specifying* what counts as a plan for agents in concurrent worlds with natural actions, we note that a regression-style planning algorithm could be based on Theorem 2, at least in the case when $LNTPC$ holds.

5.3 Example

We consider a generalization of an example that Pinto used in his Ph.D. thesis [23], which involves two natural actions and an agent's "free will". Two perfectly elastic balls, B_1 and B_2, are rolling parallel to each other on a frictionless floor, between two parallel walls. Their motions are orthogonal to the walls, so we can expect them to bounce indefinitely between the two walls, unless the agent catches one or both of them, which he is free to do. Take the first wall to be the y-axis, the second wall to be distance $W > 0$ to the right of the first wall, and the balls start their motion towards the right, beginning at the first wall. Initially, B_2 has twice the velocity of B_1.

Initial Situation

$W > 0,\ pos(B_1, S_0) = pos(B_2, S_0) = 0,\ vel(B_1, S_0) > 0,$
$vel(B_2, S_0) = 2 * vel(B_1, S_0),\ start(S_0) = 0,\ B_1 \neq B_2,$
$natural(a) \equiv (\exists t).a = bounce(B_1, t) \vee a = bounce(B_2, t).$

Action Precondition Axioms

$$Poss(bounce(b,t),s) \equiv [b = B_1 \vee b = B_2] \wedge vel(b,s) \neq 0 \wedge [vel(b,s) > 0 \supset t = start(s) + \tfrac{W - pos(b,s)}{vel(b,s)}] \wedge [vel(b,s) < 0 \supset t = start(s) - \tfrac{pos(b,s)}{vel(b,s)}].$$

$$Poss(catch(b,t),s) \equiv vel(b,s) \neq 0 \wedge \neg Poss(bounce(b,t),s).$$

Successor State Axioms

$$Poss(c,s) \supset pos(b, do(c,s)) = pos(b,s) + vel(b,s) * (time(c) - start(s)).$$

$$Poss(c,s) \supset vel(b, do(c,s)) = \textbf{if } (\exists t) catch(b,t) \in c \textbf{ then } 0 \textbf{ else if } (\exists t) bounce(b,t) \in c \textbf{ then } -vel(b,s) \textbf{ else } vel(b,s).$$

Least Natural Time Points

Using Lemma 4, and induction, we can show that:

$$lntp(s,t) \equiv Poss(bounce(B_2,t),s) \vee Poss(bounce(B_1,t),s) \wedge (\forall t') \neg Poss(bounce(B_2,t'),s).$$

It follows that $LNTPC$ holds. Notice that we have not proved that $(\forall s)(\exists t) lntp(s,t)$. In fact, this is false; A $\{catch(B_1,t), catch(B_2,t)\}$ concurrent action could intervene in some situation. This would prevent any *bounce* action from occurring in the resulting situation, so this resulting situation would have no least natural time point.

For $n = 1, 2, \ldots$ define $\tau_n = \frac{n * W}{2 * vel(B_1, S_0)}$. The τ_i are the times at which ball B_2 will bounce, assuming no $catch(B_2,t)$ actions occur. Then the following sequence of concurrent actions leads to a legal situation, provided the two actions in the concurrent actions are jointly possible:
$\{bounce(B_2,\tau_1)\}, \{bounce(B_1,\tau_2), bounce(B_2,\tau_2)\},$
$\{bounce(B_2,\tau_3)\}, \{bounce(B_1,\tau_4), bounce(B_2,\tau_4)\},$
$\{catch(B_2, [\tau_4 + \tau_5]/2)\}, \{bounce(B_1,\tau_6)\},$
$\{bounce(B_1,\tau_8)\}, \{catch(B_1,\tau_9)\}.$
This could be proved by regression, using Theorem 2, but doing so by hand would be too tedious here.

Notice that the following action, performed in S_0 may, or may not lead to a legal situation: $\{catch(B_1, \tau_1/2), catch(B_2, \tau_1/2\}$. That depends on whether or not the two catch actions are jointly possible. If the agent is a one-handed robot, then any axiomatization of the agent's abilities will include

$$(\forall c, s).Poss(c,s) \supset \neg(\exists x, y, t).x \neq y \wedge catch(x,t) \in c \wedge catch(y,t) \in c.$$

This is another instance of Pinto's precondition interaction problem. Notice that all the results of this paper are independent of any assumptions about this problem.

5.4 Discrete Time

Nothing in the previous discussion requires time to be continuous. We consider here the consequences of relaxing this assumption. Specifically, we imagine the time line to be the integers (positive and negative), so that the sort *time* now ranges over these. Notice that Zeno's paradox cannot arise in this setting. When time is discrete, we have the following:

Lemma 6 *Suppose:*

1. *The time line ranges over the integers.*
2. *The domain closure axiom (19) holds for natural actions.*
3. *Each natural action $A_i(\vec{x}, t)$ has an action precondition axiom logically equivalent to one of the form:*

$$Poss(A_i(\vec{x}, t), s) \equiv \Phi_i(\vec{x}, t, s) \wedge start(s) \leq t, \quad (23)$$

where $\Phi_i(\vec{x}, t, s)$ is a first order formula with free variables among $\vec{x}, t, s$.

Then the least natural time point condition is satisfied: Our situation calculus axioms entail $LNTPC$.

Proof: (Slightly informal)
Suppose that $\mathcal{M}$ is model of our axioms in which the *time* sort ranges over the integers, and let σ be a variable assignment such that

$$\mathcal{M}, \sigma \models natural(a) \wedge Poss(a, s).$$

We prove that $\mathcal{M}, \sigma \models (\exists t) lntp(s, t)$. By axiom (4), domain closure (19), and the assumption (23),

$$\mathcal{M}, \sigma \models start(s) \leq time(a).$$

So, for any natural action a, $time(a)$ is bounded from below in $\mathcal{M}$ by $start(s)$. Since time is discrete, there is a least $t \geq start(s)$ for which $\mathcal{M}, \sigma \models (\exists a).natural(a) \wedge Poss(a, s) \wedge time(a) = t$. Hence, $\mathcal{M} \models LNTPC$.

□

The above lemma is actually more general than it initially appears to be. To begin, without some kind of domain closure assumption on natural actions, it is impossible to prove the legality of any situation. Secondly, it is quite natural to impose the temporal constraint $start(s) \leq t$ on action precondition axioms, as in (23), or, as we did in the bouncing balls example, omit this constraint from the axioms when it is known from the problem description that the time variable t necessarily satisfies this constraint.

6 Discussion and Conclusions

By basing it on the language $\mathcal{A}$ of Gelfond and Lifschitz [6], Baral and Gelfond [3] provide a semantic account of concurrency which, although not formulated in the situation calculus, has many similarities with ours. The principal difference is that Baral and Gelfond focus exclusively on concurrency, so their ontology does not include time or natural actions. Moreover, $\mathcal{A}_C$, their action representation language, is propositional; while it would be possible to translate $\mathcal{A}_C$ theories into the situation calculus, the resulting sentences would be in the *monadic* situation calculus, and therefore would be less general than the logical theories to which our approach applies.

There have been a few earlier papers on formalizing natural actions and continuous time. Shanahan's approach [30] is embedded in the *event calculus* (Kowalski and Sergot [11]); Sandewall [27] relies on a temporal logic. Accordingly, these proposals are difficult to compare with ours, based as it is on the situation calculus. Below, we provide a comparison along one dimension: abductive planning, which seems to be required by these proposals, and the deductive planning approach of the situation calculus.

The approaches of Pinto [23] and Pinto and Reiter [24], and of Miller and Shanahan [20] come closest to that of this paper in that they also rely on the situation calculus. These all differ from us in proposing something like an "actual" path in the tree of situations, corresponding to the way in which the world actually evolves. Both proposals suffer from what might be called the "premature minimization problem", which amounts to the assumption that all action occurrences (natural as well as those under the free will of an agent) are either specified as part of the axiomatization, or are inferable from it. Closure, in the form of the minimization of action occurrences, is enforced by suitably circumscribing these axioms. This means that any agent initiated action not deducible from the axioms is assumed not to have happened. Therefore, if A is an action which the agent could have, but did not initiate according to the axioms, then the "actual" path of world actions will not include A. In other words, *hypothetical* world evolutions, in which all possible agent initiated actions are permitted, are excluded from the minimized axioms. The only legal situations (in the sense of this paper) under these approaches are those on the actual path. Now one of the great advantages of the situation calculus is that all possible world histories are explicitly available in the language, as object language terms.[4] It is precisely this feature which permits a *deductive* approach to hypothetical reasoning about possible world futures. This, in turn, pro-

[4] That is the role of the function symbol *do*. Like *cons* in LISP, it creates sequences, in this case terms representing world histories, i.e. sequences of actions.

vides for a deductive account for planning. Unfortunately, by prematurely minimizing action occurrences, the above approaches preclude a deductive approach to hypothetical reasoning and planning; analogous versions of the definition of planning of Section 5.1 cannot be given. Instead, an *abductive* account must be used. Intuitively, one can see why this must be so. Since, after circumscribing the axioms, there is just one actual path, the possibility of other actual paths can be considered only by hypothetically postulating other free will action occurrences, closing the axioms with respect to these hypothetical occurrences, and testing whether the resulting axioms are satisfiable and entail the goal condition.

This phenomenon of planning by abduction is quite widespread; it is used, for example, in the event calculus [5] and in Allen's temporal logic [1]. In fact, it is the only way to do planning in logics which do not provide for branching futures. Unfortunately, abductive planning suffers from a number of drawbacks, when compared with the deductive approach:

1. It is a metalevel task; the planner must leave the object language to generate a candidate collection of atoms of the form $occurs(A, T)$, test the consistency of these atoms with respect to the object level axiomatization of the domain, then return to the object level to prove the goal sentence relative to the enlarged axiom set.
2. Because of the above consistency test, abductive plans are not even recursively enumerable for first order axiomatizations, in contrast to the deductive case.
3. Even if we ignore the noncomputability of the consistency test, from a computational point of view there are at least two theorem proving tasks for abductive planners: the consistency test (which normally must be performed several times), together with the goal-entailment proof.
4. It is not difficult to imagine settings where a robot agent needs to establish that a plan does *not* exist. In the deductive case, this amounts to establishing that $Axioms \models \neg(\exists s)G(s)$, and, at least formally, is no more problematic than the planning problem. For abductive planners, it is not at all obvious what such a proof might look like, or how it could be constructed; the robot must show, again at the metalevel, that there is no finite set of atoms of the form $occurs(A, T)$, consistent with the background axioms, which entails the goal.
5. As observed by Pelavin [22], for concurrent actions, abductive planning can yield incorrect plans in the presence of partial world descriptions.

Pelavin [22] addresses the formalization of concurrent actions by extending the ontology of Allen's linear time logic [1] to include histories to represent branching futures, and suitable modal operators semantically characterized with respect to these histories. This allows a deductive account of planning within a temporal logic, but at the expense of a rather complicated logic.

Acknowledgements:

Thanks to the other members of the University of Toronto Cognitive Robotics Group (Yves Lespérance, Hector Levesque, Fangzhen Lin, Daniel Marcu and Richard Scherl) for their comments and suggestions. This paper was strongly influenced by Javier Pinto's approach to concurrency and natural actions in his Ph.D. thesis. I have also benefited from extensive discussions about temporal reasoning with Javier, and with Mikhail Soutchanski and Eugenia Ternovskaia. Rob Miller provided useful comments on an earlier draft of this paper, and Rob and Murray Shanahan helped me better understand their approach to natural actions in the situation calculus. Todd Kelley first pointed out to me the importance of accommodating enabling actions; this motivated the current form of the foundational axiom (10) which now allows such actions to be easily represented. Hesham Khalil carefully read an earlier draft of this paper, and made many useful suggestions; in particular, he noted the need for the foundational axiom (7). 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.

References

[1] J.F. Allen. Temporal reasoning and planning. In J.F. Allen, H.A. Kautz, R.N. Pelavin, and J.D. Tenenberg, editors, *Reasoning about Plans*, pages 1–68. Morgan Kaufmann Publishers, San Francisco, CA, San Mateo, CA, 1991.

[2] F. Bacchus, J.Y. Halpern, and H.J. Levesque. Reasoning about noisy sensors in the situation calculus. In *Proc. IJCAI'95*, pages 1933–1940, 1995.

[3] C. Baral and M. Gelfond. Reasoning about effects of concurrent actions. *Journal of Logic Programming*, 1996. to appear.

[4] Ernest Davis. Infinite loops in finite time. Technical report, Department of Computer Science, New York University, February, 1992.

[5] K. Eshghi. Abductive planning with event calculus. In *Proceedings of the Fifth International Conference on Logic Programming*, pages 562–579. MIT Press, 1988.

[6] M. Gelfond and V. Lifschitz. Representing actions in extended logic programs. In *Proc. Joint Int. Conf. and Symp. on Logic Programming*, pages 559–573, 1992.

[7] 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.

[8] C.C. Green. Theorem proving by resolution as a basis for question-answering systems. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 183–205. American Elsevier, New York, 1969.

[9] A. R. Haas. 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, Los Altos, California, 1987. Morgan Kaufmann Publishers, San Francisco, CA.

[10] T.G. Kelley. Modeling complex systems in the situation calculus: A case study using the Dagstuhl steam boiler problem. 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)*. Morgan Kaufmann Publishers, San Francisco, CA, 1996.

[11] R.A. Kowalski and M.J. Sergot. A logic-based calculus of events. *New Generation Computing*, 4:267, 1986.

[12] 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 Symposium*, November, 1994. New Orleans, LA.

[13] H.J. Levesque. What is planning in the presence of sensing? In *Proceedings of the National Conference on Artificial Intelligence*, 1996. To appear.

[14] H.J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. Scherl. GOLOG: a logic programming language for dynamic domains. *Journal of Logic Programming, Special Issue on Actions*, 1996. To appear.

[15] F. Lin. Specifying the effects of indeterminate actions. In *Proceedings of the National Conference on Artificial Intelligence*, 1996. To appear.

[16] F. Lin and R. Reiter. State constraints revisited. *J. of Logic and Computation, special issue on actions and processes*, 4:655–678, 1994.

[17] F. Lin and Y. Shoham. Concurrent actions in the situation calculus. In *Proceedings of the National Conference on Artificial Intelligence*, pages 590–595, 1992.

[18] D. Marcu, Y. Lespérance, H. Levesque, F. Lin, 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 Volume II – Proceedings of the 1995 Workshop on Agent Theories, Architectures, and Languages (ATAL-95)*, pages 331–346. Springer-Verlag, Lecture Notes in Artificial Intelligence, 1996. To appear.

[19] 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.

[20] R. Miller and M. Shanahan. Narratives in the situation calculus. *The Journal of Logic and Computation (Special Issue on Actions and Processes)*, 4:513–530, 1994.

[21] E.P.D. Pednault. ADL: Exploring the middle ground between STRIPS and the situation calculus. In R.J. Brachman, H. Levesque, and R. Reiter, editors, *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning (KR'89)*, pages 324–332. Morgan Kaufmann Publishers, San Francisco, CA, 1989.

[22] R.N. Pelavin. Planning with simultaneous actions and external events. In J.F. Allen, H.A. Kautz, R.N. Pelavin, and J.D. Tenenberg, editors, *Reasoning about Plans*, pages 127–211. Morgan Kaufmann Publishers, San Francisco, CA, San Mateo, CA, 1991.

[23] J.A. Pinto. *Temporal Reasoning in the Situation Calculus*. PhD thesis, University of Toronto, Department of Computer Science, 1994.

[24] J.A. Pinto and R. Reiter. Adding a time line to the situation calculus. In *Second Symposium on Logical Formalizations of Commonsense Reasoning*, pages 172–177, Austin, Texas, Jan. 11-13, 1993.

[25] 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.

[26] R. Reiter. Proving properties of states in the situation calculus. *Artificial Intelligence*, 64:337–351, 1993.

[27] E. Sandewall. Combining logic and differential equations for describing real-world systems. In R.J. Brachman, H. Levesque, and R. Reiter, editors, *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning (KR'89)*, pages 412–420. Morgan Kaufmann Publishers, San Francisco, CA, 1989.

[28] R. Scherl and H.J. Levesque. The frame problem and knowledge producing actions. In *Proc. AAAI-93*, pages 689–695, Washington, DC, 1993.

[29] L.K. Schubert. Monotonic solution of the frame problem in the situation calculus: an efficient method for worlds with fully specified actions. In H.E. Kyberg, R.P. Loui, and G.N. Carlson, editors, *Knowledge Representation and Defeasible Reasoning*, pages 23–67. Kluwer Academic Press, 1990.

[30] M.P. Shanahan. Representing continuous change in the event calculus. In *Proceedings ECAI 90*, pages 598–603, 1990.

[31] S. Shapiro, Y. Lespérance, and H.J. Levesque. Goals and rational action in the situation calculus – a preliminary report. In *Working Notes of the AAAI Fall Symposium on Rational Agency: Concepts, Theories, Models and Applications*, Cambridge, MA, 1995.

[32] E. Ternovskaia. Interval situation calculus. In *Proc. of ECAI'94 Workshop W5 on Logic and Change*, pages 153–164, Amsterdam, August 8-12, 1994.

[33] R. Waldinger. Achieving several goals simultaneously. In E. Elcock and D. Michie, editors, *Machine Intelligence 8*, pages 94–136. Ellis Horwood, Edinburgh, Scotland, 1977.

Only Knowing in the Situation Calculus

Gerhard Lakemeyer
Institute of Computer Science III
University of Bonn
Römerstr. 164
D-53117 Bonn, Germany
gerhard@cs.uni-bonn.de

Abstract

Actions depend crucially on what an agent knows and does not know. For example, an action may have a precondition that requires knowing the referent of a term, which is generally referred to as *knowing who* or *knowing what*. Alternatively, executing a sense action may be the result of realizing that the referent of a term is not known yet. The latter requires an agent to reason about all it knows about the world. This concept, also called *only knowing*, has been studied using possible-world semantics, yet only in the static case. One of the best understood action formalisms is the situation calculus. Moreover, it also comes equipped with a possible-world model of knowledge, which has led to deep insights into the relationship between action and knowledge. Adding only knowing to the situation calculus, which is the topic of this paper, turns out to be much more problematic than in the case of adding knowledge. It requires a reconstruction of the situation calculus itself by first developing a possible-world model of action and then interpreting situations explicitly as possible worlds. The properties of the situation calculus, which normally need to be stipulated axiomatically, are shown to be valid formulas in our model. More importantly, only knowing is fully integrated into the action formalism.

1 Introduction

Actions depend crucially on what an agent knows and does not know. For example, an action may have a precondition that requires knowing the referent of a term, which is generally referred to as *knowing who* or *knowing what*. Alternatively, executing a sense action may be the result of realizing that the referent of a term is not known yet. For instance, consider a robot in an office environment. The action *goto*(*Room_A*) is possible only if the robots knows *Room_A*. Logically, this means that the robot has to know a standard name for *Room_A*. In practice, this could mean knowing its coordinates in a metric occupancy map of the environment.[1] A complex action like *visit_all_rooms*, translating into a sequence of *goto*'s, would require, in addition, knowing *all* rooms. Now suppose the robot is put in an unknown office floor and is told that there are two rooms, A and B (perhaps by supplying a topological map of the floor). Initially the metric coordinates of A and B are unknown, that is, actions like *goto*(A) or *visit_all_rooms* would fail. Let us assume the robot can perform *sense_room*(x), which returns the location of x on the occupancy map.[2] A knowledge level specification of how to determine the standard names of all the rooms would be: *While there is a room x which you do not know yet,* sense_room*(x)*. After that the robot knows all the rooms, that is, *goto*(A) and *visit_all_rooms* are now possible.

Such high level specifications require a model that integrates both action and knowledge in an appropriate way. Perhaps the best developed model of this kind is due to Moore [Moo85a], which was later extended by Scherl and Levesque [SL93]. They analyze knowledge and action by axiomatizing possible-world semantics, a well understood model of knowledge and belief [Kri63, Hin62, HM92], within the situation calculus [McC63], which has received renewed attention recently (e.g., [Rei91, Pin94, BHL95]). Let us focus at this point on representing knowledge in the situation calculus. (A full account of the language is given in Section 4.) First, the language allows us to express what is true at a given situation. For example, in or-

[1] Occupancy maps are a favorite method of spatial representations in robotics [Mor88, BBC95].

[2] In practice, this might require exploring the environment, constructing an occupancy map on the fly [BBC95]. Having finally found the room with the help of the topological information, the corresponding coordinates on the occupancy map are returned.

der to say that $\alpha = (\forall x Room(x) \equiv (x = A)$ holds in situation s we would write $\alpha[s] = (\forall x Room^{fl}(x, s) \equiv (x = A^{fl}(s))$, where $Room^{fl}$ and $A^{fl}(s)$ are so-called *fluents*. Knowledge is introduced by importing from possible-world semantics the idea of epistemic alternatives for a given situation, that is, an agent considers a number of situations possible. An agent then knows a formula just in case it is true in all these situations. To formalize this idea, a special two-place fluent K (which we also call K-fluent) is introduced. $K(s', s)$ means that the situation s' is (epistemically) accessible from s. Knowing α at s, which we denote as $(\mathbf{K}\alpha)[s]$, can then be expressed as $\forall s' K(s', s) \supset \alpha[s']$.

Let us assume the language includes standard names, that is, knowing what A is amounts to knowing $(A = n)$ for some standard name n. In the above example, if α is all the information the robot has about situation s, it should be able to conclude that $\mathbf{K}(\exists x Room(x) \wedge \neg \mathbf{K} Room(x))[s]$,[3] that is, it knows that there is a room which is yet unknown. However, this does not follow from $(\mathbf{K}\alpha)[s]$ even if the K-fluent allows full introspection[4] simply because it is not ruled out that $(A = n)$ indeed holds at all situations accessible from s. In order to get the right conclusion we need to use the assumption that α is all that is known (or *only known*). Levesque [Lev90] gave *only knowing* a very simple possible-world interpretation. In a nutshell, α is only known if it is known and, in addition, any world that satisfies α is itself accessible. Superficially it seems this would be easy to capture in the situation calculus using $\forall s' K(s', s) \equiv \alpha[s']$ (*) as the definition. This fails, however, because Levesque's approach requires that there are "enough" worlds in the model to choose from (in essence, one world for each maximally consistent set of first-order sentences). Given (*), it is still possible that all situations assign the same standard name to A. There does not seem to be an easy fix to the problem. One might be tempted to stipulate the existence of enough situations by adding an infinite number of appropriate axioms. However, that could never match the uncountable number of worlds in Levesque's case and it is unclear what logic would result, never mind proving properties about it. (In Section 4, it will become clear that (*) has even more problems because it may force situations to be accessible that have different action histories.)

Our solution to the problem is to move away from merely emulating possible-world semantics in the situation calculus. Instead, we will adopt Levesque's possible-world model directly, enrich it with a model of action, and then interpret the situation calculus with respect to that semantics. In particular, situation variables will be allowed to range over all possible worlds. Since the model has enough worlds, a definition very close to (*) will give us the desired concept of only knowing as part of the situation calculus. The main technical results are: (a) our semantic reconstruction of the situation calculus is faithful in that the usual properties come out as valid formulas in our model and (b) Levesque's logic of only knowing is completely embedded in a precise sense in this version of the situation calculus.

In related work, del Val and Shoham [dVS94], as far as we know, provide the only other approach to only knowing in the situation calculus. They do not use a K-fluent but axiomatize belief directly using a function *bel* whose argument is a term denoting a propositional formula. In their approach α is only known at a situation if any sentence that is known at that situation is also known at any other situation where α is known. In addition they need to assume that any consistent formula is known at some situation. While they obtain reasonable properties for only knowing this way, the approach is limited since it applies only to propositional beliefs and no beliefs about beliefs are allowed. It is not at all clear how to generalize this framework to the full first-order case with nested beliefs and quantifying-in as considered in this paper.

The rest of the paper is organized as follows. In Section 2, we introduce Levesque's logic of only knowing. Section 3 adds a theory of action to the underlying possible-world semantics. In Section 4, we introduce a variant of the situation calculus which is interpreted with respect to the model of knowledge and action developed in the previous section. Section 5 ends the paper with a brief summary and pointers to future work.

2 The Logic *OL*

The language $\mathcal{L}$ is a modal first-order dialect with equality and function symbols plus a countably infinite set of standard names $\mathcal{N} = \{\#1, \#2, \ldots\}$. We denote the set of all variables by $\mathcal{V}$. *Terms* are defined as usual. Similarly, the *formulas* of $\mathcal{L}$ are constructed in the standard way from the atomic formulas, the connectives $\neg$ and $\vee$, the quantifier $\forall$,[5] and the modal operators $\mathbf{K}$ and $\mathbf{O}$, where $\mathbf{K}\alpha$ should be read as "the agent knows α" and $\mathbf{O}\alpha$ 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.

Notation: Sequences of terms or variables are often written in vector notation. For example, a sequence of variables $\langle x_1, \ldots, x_k \rangle$ is abbreviated as $\vec{x}$. Also $\forall \vec{x}$

[3] short for $\forall s' K(s', s) \supset \exists x Room(x, s') \wedge \neg(\forall s'' K(s'', s') \supset Room(x, s''))$.

[4] This is the case if K is a transitive and Euclidean relation [HM92].

[5] Other logical connectives like $\wedge$, $\supset$, and $\equiv$ and the quantifier $\exists$ are used freely and are defined in the usual way.

stands for $\forall x_1 \ldots \forall x_k$.

The semantics of objective formulas is determined by *worlds*, which are ordinary first-order structures with the restriction that all worlds have as their domain of discourse the set of standard names $\mathcal{N}$, which is a simple way of guaranteeing that standard names have their desired meaning.[6] Formally, a world w maps every n-ary predicate symbol into an n-ary relation over $\mathcal{N}^n$, and every n-ary function symbol into an n-ary function from $\mathcal{N}^n$ into $\mathcal{N}$. Since we later modify the meaning of a world, we also use the term *valuation* for this basic form.

A *variable map* ν is a mapping from the variables into $\mathcal{N}$. Given a ν, a variable x, and $n \in \mathcal{N}$, ν_n^x is the same variable map as ν except that x is mapped into n. Given a world w and a variable map ν, the denotation of an arbitrary term t is defined as

$|n|_{w,\nu} = n$, where n is a standard name.
$|x|_{w,\nu} = \nu(x)$, where x is a variable.
$|f(t_1, \ldots, t_n)|_{w,\nu} = H(n_1, \ldots, n_n)$, where $H = w(f)$ and $n_i = |t_i|_{w,\nu}$.

For sequences of terms $\vec{t} = \langle t_1, \ldots, t_k \rangle$ we often write $|\vec{t}|_{w,\nu}$ instead of $\langle |t_1|_{w,\nu}, \ldots, |t_k|_{w,\nu} \rangle$.

While the truth of objective formulas is determined by *worlds*, the meaning of formulas of the form $\mathbf{K}\alpha$ and $\mathbf{O}\alpha$ is defined relative to a set of worlds e. We assume that the real world is always part of e. $\mathbf{K}\alpha$ holds if α is true in all worlds of e. $\mathbf{O}\alpha$ holds if $\mathbf{K}\alpha$ 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 referred to as a *state*.

The semantic rules which determine the truth of a formula α at a given world w, epistemic state e, and variable map ν (denoted as $e, w, \nu \models \alpha$), are defined as follows.

$$
\begin{array}{lcl}
e,w,\nu \models \mathrm{P}(\vec{t}) & \iff & \vec{n} \in w[P] \text{ and } \vec{n} = |\vec{t}|_{w,\nu}, \\
 & & \text{where } P(\vec{t}) \text{ is atomic.} \\
e,w,\nu \models t_1 = t_2 & \iff & |t_1|_{w,\nu} = |t_2|_{w,\nu} \\
e,w,\nu \models \neg\alpha & \iff & e,w,\nu \not\models \alpha \\
e,w,\nu \models \alpha \vee \beta & \iff & e,w,\nu \models \alpha \text{ or } e,w,\nu \models \beta \\
e,w,\nu \models \forall x\alpha & \iff & e,w,\nu_n^x \models \alpha \text{ for all } n \in \mathcal{N} \\
e,w,\nu \models \mathbf{K}\alpha & \iff & \text{for all } w', \\
 & & \text{if } w' \in e, \text{ then } e,w',\nu \models \alpha \\
e,w,\nu \models \mathbf{O}\alpha & \iff & \text{for all } w', \\
 & & w' \in e \text{ iff } e,w',\nu \models \alpha
\end{array}
$$

A formula α is valid ($\models_{\mathrm{OL}} \alpha$) iff $e, w, \nu \models \alpha$ for all worlds w, all sets of worlds e containing w, and all variable maps ν.[7] We sometimes write $e, w \models \alpha$ if α is a sentence and $w \models \alpha$ if α is an objective sentence.

For a detailed discussion of *OL*, we refer the reader to [Lev90]. Here we only note that the operator $\mathbf{K}$ has the usual properties of the logic $S5$ [HC68, HM92] whose characteristic axioms are:

K: $\mathbf{K}(\alpha \supset \beta) \supset (\mathbf{K}\alpha \supset \mathbf{K}\beta)$
T: $\mathbf{K}\alpha \supset \alpha$
4: $\mathbf{K}\alpha \supset \mathbf{K}\mathbf{K}\alpha$
5: $\neg\mathbf{K}\alpha \supset \mathbf{K}\neg\mathbf{K}\alpha$

In addition, the Barcan formula $\forall x\mathbf{K}\alpha \supset \mathbf{K}\forall x\alpha$ is also valid since we are assuming a fixed universe of discourse [HC68]. While $\mathbf{K}$ is very well understood, this is less the case for $\mathbf{O}$ except perhaps when $\mathbf{O}$ is applied to an objective sentence. For consider an atomic sentence p. It is easy to see that the only epistemic state e where $\mathbf{O}p$ 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 $\mathbf{O}$, which has the effect of maximizing e. As a result, the only objective sentences known at e are the logical consequences of p, which most people agree captures that "p is all that is known" very well. The story of $\mathbf{O}$ becomes much more complicated when allowing arbitrary formulas in the scope of $\mathbf{O}$. The interested reader is referred to [Lev90], which includes a complete axiomatization of $\mathbf{O}$ in the propositional case. Its axiomatization in the first-order case is still an open problem [HL95]. Fortunately, the ideas presented in this paper will not require the reader to contemplate $\mathbf{O}$ when applied to nonobjective formulas.

3 Actions and Possible Worlds

So far, a model of *OL* presents a static view of the world and the agent's knowledge. We now want to extend the semantics by adding actions that are able to change the world and what the agent knows about it. Many of the following ontological decisions as far as actions and their effects on worlds go directly reflect those that are usually made in the situation calculus.

Before going into the technical details let us look at the approach and underlying assumptions informally. First, we distinguish between *physical actions* such as dropping a ball and *sense actions* like reading the combination of a safe from a piece of paper. While the former change the state of the world as well as the agent's knowledge, the latter can be thought of as leaving the world as is and changing only the agent's epistemic state.[8]

Performing a physical action from a possible-world perspective merely means that we move from the current world to a new world which reflects the effects of the action. As far as the agent's knowledge goes, we

[6] See [Lev84] for more details.

[7] In contrast to Levesque, we require knowledge instead of belief, that is, the real world is always a member of the worlds considered possible. Also, Levesque used *maximal sets* to define validity, a complication we ignore here for simplicity.

[8] In Moore [Moo85a], physical and sense actions were called uninformative and informative actions, respectively.

make the assumption, as is usually done in the situation calculus as well, that the agent has perfect control over all the actions and hence has perfect knowledge about their effects. Now recall that the intuition behind the worlds in the agent's epistemic state is that, as far as the agent is concerned, any of them could be the real world. Hence it seems natural to assume that the new epistemic state that results from a physical action is obtained by "mentally" performing the action in each of the worlds of the old state.

To see how sense actions work, let us assume the agent wants to sense the truth value of an atomic formula P. After the action nothing should have changed except that the agent now knows whether P is true or not. In other words, the agent should remove all those worlds from its epistemic state which conflict with the actual truth value of P. Similarly, sensing the referent of a term leaves the agent in a state where it knows the standard name of the term.

As in the situation calculus, we assume that all actions have names and they can be applied to domain objects such as *drop*(*ball*). We assume a countably infinite set of action symbols of every arity partitioned into physical and sense action symbols. We assume that sense actions have the form A^{ϕ} for every sentence and closed term ϕ in $\mathcal{L}$. Given an n-ary action symbol A and standard names $n_1, \ldots n_k$, $A(n_1, \ldots, n_k)$ is called a *primitive action*. We identify the set of all actions with the set of all primitive actions, which we denote by $\mathcal{A}$. In other words, primitive actions will be like standard names in that they uniquely identify actions.

Before formalizing actions in possible worlds, we need to address another assumption usually made in the situation calculus, namely that performing an action always leads to a new world/situation never seen before. Given that worlds so far are defined extensionally, this would not work for sense actions which we said do not change what is true in a world. To overcome this difficulty, we add more structure to worlds by adding to them an *action log* which keeps track of the initial world and all the primitive actions which have since led to the world at hand. So even if an action A does not change the truth value of any atomic formula, we will get a new world simply because the action log gets updated by A. Formally a world is now a tuple $\langle \omega, \lambda \rangle$, where ω is a valuation (or world in the old sense) and λ is an *action log* of the following kind.

Definition 1 *The set of action logs Λ is defined as the smallest set which contains*

1. $\langle \omega \rangle$ *for every valuation* ω.
2. *If λ is in Λ, then so is $\lambda|A$, where $A \in \mathcal{A}$ and $\lambda|A$ is λ with A appended to the end of the sequence.*

Intuitively, an action log $\omega A_1 \ldots A_n$ means that starting at a world with valuation ω, the actions $A_1, \ldots, A_n$ have been performed, in that order.[9]

Let $\mathcal{W}$ to be set of all worlds in this new sense. A world $\langle \omega, \lambda \rangle$ is called an *initial world* if $\lambda = \langle \omega \rangle$. In other words, initial worlds are the starting points for all actions to come. They will serve as the denotation of the initial situation S_0 once we add the situation calculus to our language. Given an initial world w and a set of initial worlds e with $w \in e$, (e, w) is called an *initial state*.

Although we in general allow states that contain worlds with action logs whose sequences of actions differ, we are really only interested in states which evolve from initial states by applying actions to them. These states will have the property that the worlds contained in them all have the same action sequence in their action logs. Similarly, we allow worlds $w = \langle \omega, \langle \omega' A_1 \ldots A_n \rangle \rangle$, where the valuation ω does not match what is predicted by the action log. Again, we are really only interested in worlds evolving from initial worlds, and these will always have matching valuations and action logs by design.

From now on we will use the term *world* only in sense just defined. Whenever we need to refer to the semantics of *OL* of the previous section, we will from now on only use the term *valuation*. As we will see later, it is sometimes useful to convert worlds into valuations by stripping off their action logs or to turn valuations into initial worlds.

Definition 2 *For any valuation w let $w\uparrow = \langle w, \langle w \rangle \rangle$ be the corresponding initial world. Also, for a set of valuations e, let $e\uparrow = \{w\uparrow \mid w \in e\}$.*

Similarly for any world $w = \langle \omega, \lambda \rangle$, let $w\downarrow = \omega$ and for any set of worlds e, let $e\downarrow = \{w\downarrow \mid w \in e\}$ be the corresponding set of valuations.

To model how an action changes the state of a world we introduce an action execution function *Do*, which determines what the world looks like after an action is performed, and an action precondition function *poss*, which tells us when it is possible to execute an action.

$$poss : \mathcal{A} \times \mathcal{W} \longrightarrow \{0, 1\}.$$
$$Do : \mathcal{A} \times \mathcal{W} \longrightarrow \mathcal{W}.$$

Do is required to satisfy the following properties.

Let $Do(A, \langle \omega, \lambda \rangle) = \langle \omega', \lambda' \rangle$. Then

1. $\lambda' = \lambda|A$ and
2. if A is a sense action or $poss(A, w) = 0$, then $\omega' = \omega$.

The first restriction simply states that performing an action in a world leads to a new world with a correctly

[9] Interestingly, a similar concept was recently proposed independently in [BHL95].

updated action log. The second makes sure that a sense action or an action whose execution is impossible does not change what is true in the world.[10]

We also need to specify how the epistemic state of an agent changes when performing an action in a world w. For that purpose we introduce a function

$$Do^e : \mathcal{A} \times 2^{\mathcal{W}} \times \mathcal{W} \longrightarrow 2^{\mathcal{W}},$$

which distinguishes between physical and sense actions. In the case of a physical action A and an epistemic state e, Do^e is completely determined by Do since the action is simply performed locally in every world of e:

$$Do^e(A, e, w) = \{w^* \mid \exists w^{**} \in e \text{ and } w^* = Do(A, w^{**})\}$$

Let us now consider the case of a sense action A^ϕ, where ϕ is a sentence in $\mathcal{L}$. Performing A^ϕ at e means that the agent senses the truth value of ϕ. In other words, after doing A^ϕ, the agent should know whether ϕ was true at e and w. Should the sense action be impossible to execute, then Do^e will not change the epistemic state except that we record the action in the action log.

$$Do^e(A^\phi, e, w) = \begin{cases} \{w^* \mid \exists w^{**} \in e, w^* = Do(A^\phi, w^{**}) \\ \text{and } e{\downarrow}, w^{**}{\downarrow} \models \phi \text{ iff } e{\downarrow}, w{\downarrow} \models \phi.\} \\ \qquad\qquad \text{if } poss(A^\phi, w) = 1 \\ \{w^* \mid \exists w^{**} \in e \text{ and} \\ w^* = Do(A^\phi, w^{**})\} \qquad \text{otherwise.} \end{cases}$$

Note that $Do(A^\phi, w^{**})$ only has the effect of updating the action log by A^ϕ. In effect, the agent learns not only the truth value of ϕ but also that she has performed A^ϕ by using w^* instead of w^{**}.

Similarly, if ϕ is a closed term, that is, A^ϕ senses the referent of ϕ, we obtain:

$$Do^e(A^\phi, e, w) = \begin{cases} \{w^* \mid \exists w^{**} \in e, w^* = Do(A^\phi, w^{**}) \\ \text{and for all } n \in \mathcal{N}, w^{**}{\downarrow} \models (\phi = n) \\ \text{iff } w{\downarrow} \models (\phi = n).\} \\ \qquad\qquad \text{if } poss(A^\phi, w) = 1 \\ \{w^* \mid \exists w^{**} \in e \text{ and} \\ w^* = Do(A^\phi, w^{**})\} \qquad \text{otherwise.} \end{cases}$$

We are now ready to define what we mean by an action model.

Definition 3 *$\mathcal{M} = \langle e, w, Do, poss\rangle$ is called an* action model *if (e, w) is an initial state, and Do and $poss$ are action execution and precondition functions, respectively.*

[10] If an action is impossible, an alternative approach would be to leave Do undefined in that case. While this might be considered more intuitive, we have opted for a totally defined Do, since it leads to a simpler model. One reading of our model would be that the (failed) attempt to execute the action is nevertheless recorded in the action log.

The following definitions will be useful in the next section when we introduce the language of the situation calculus to talk about action models.

Definition 4 *A world w' is* reachable *from a world w iff $w = w'$ or $w' = Do(A, w^*)$ for some primitive action A and world w^*, and w^* is reachable from w.*

We say that a state (e', w') is reachable from a state (e, w) iff either $e' = e$ and $w' = w$ or there is a (e^*, w^*) and a primitive action A such $e' = Do^e(A, e^*, w^*)$, $w^* = Do(A, w)$, and (e^*, w^*) is reachable from (e, w). Sometimes we only care about e' being reachable. Hence we say that e' is reachable from (e, w) iff there is a world w' such that (e', w') is reachable from (e, w).

Definition 5 *Given an action model $\mathcal{M}$ and worlds w' and w, w' is* strongly reachable *from w iff there is a world w^* and a primitive action A such that $w' = Do(A, w^*)$, $poss(A, w) = 1$, and either $w = w^*$ or w^* is strongly reachable from w.*

In other words, while reachability merely means that we can get from one state to another by performing a finite number of primitive actions, strong reachability requires, in addition, that each action is indeed executable.

Definition 6 *Two worlds w and w' are called* comparable *($w_1 \approx w_2$) iff their actions logs are identical except possibly for the first element of the logs (the initial worlds).*

4 $OLS = OL$ plus situations

At this point, we can model what effect actions have on a given world and epistemic state, but the agent itself cannot reason about it. In order to do so we introduce the language of the situation calculus as the meta language to talk and reason about the dynamics of the system. Note that, in contrast to the usual use of the situation calculus where properties of situations and actions are stipulated as proper axioms of a first order theory, our approach is completely semantic in that the meaning of situations and actions is entirely determined by the action models defined in the previous section.

Syntax: The language $\mathcal{L}_s$ extends $\mathcal{L}$ by adding the usual situation calculus constructs (plus a few others). As is customary in the situation calculus, $\mathcal{L}_s$ is a sorted language with three sorts: actions, situations, and objects. The latter are just the standard names in $\mathcal{L}$. The variables $\mathcal{V}$ of $\mathcal{L}$ are now precisely those of type object, and we introduce new variables for both actions and situations. Object, situation and action terms are defined as follows:

Definition 7

1. *The set of all object terms is the set of all terms inherited from $\mathcal{L}$.*
2. *The set of action terms is the set of all action variables together with the set of all $A(t_1, \ldots, t_n)$, where A is an action symbol and the the t_i are object terms.*
3. *The set of situation terms is the smallest set such that*
   (a) *Every situation variable is a situation term.*
   (b) *The special constant S_0 is a situation term.*
   (c) *If σ is a situation term, A an action symbol, and $\vec{t}$ are terms of type object, then $do(A(\vec{t}), \sigma)$ is a situation term.*

In addition to the functions and predicates of $\mathcal{L}$, whose arguments are restricted to be of type object, we introduce for every n-ary predicate P and function symbol f fluent counterparts P^{fl} and f^{fl} of arity $n+1$, whose last argument is of type situation. Equality is extended to have arguments of all types with the restriction that both arguments have to be of the same type. We introduce special predicates $Poss(a, s)$, $s < s'$, $K(s', s)$, and $Comp(s, s')$. The first three have their usual situation calculus connotations. $Poss(a, s)$ expresses that it is possible to perform the action a in situation s. $s < s'$ tells us that s' is strongly reachable from s, that is, a sequence of actions, which are all possible, leads from s to s'. Finally, $K(s', s)$ is the K-fluent denoting epistemic alternatives as discussed already in the introduction. *Comp* is new and determines whether two situations are comparable in the sense of being reachable from (possibly different) initial situations by the same sequence of actions. *Comp* will be needed when we model only knowing in *OLS*.

In addition to the quantifiers in $\mathcal{L}$ we introduce a new universal quantifier $\forall^r$, which may be applied only to situation variables. (We also allow the regular $\forall$ to apply to situation variables.) The purpose of $\forall^r$ is to allow us to restrict quantification to those situations *reachable* from S_0.

Given these constructs, the formulas of $\mathcal{L}_s$ are obtained in the usual way.

Semantics: The semantics is defined over action models. As far as formulas of $\mathcal{L}$ are concerned, the semantics is the same as for *OL* except that we need to adapt it to our new notion of a world. Regarding situation and action terms we do not extend the universe of discourse to give meaning to them. Instead we use their natural counterparts in the action models, namely primitive actions and worlds, directly to define their denotation.

First we extend variable maps ν to take care of all variables of the extended language, that is, a variable map ν is redefined to apply to object, action, and situation variables such that.

1. $\nu(x) \in \mathcal{N}$ for each object variable x.
2. $\nu(a) \in \mathcal{A}$ for each action variable a.
3. $\nu(s) \in \mathcal{W}$ for each situation variable s.

Given a world $w = \langle \omega, \lambda \rangle$ and a variable map ν, the denotation of object terms, i.e. terms of $\mathcal{L}$, is given by $|t|_{\omega,\nu}$ as defined in Section 2. However, we will continue to write $|t|_{w,\nu}$ instead of $|t|_{\omega,\nu}$.

The denotation of an arbitrary action term t_a is defined as follows:

$$|t_a|_{w,\nu} = \begin{cases} \nu(t_a) & \text{if } t_a \text{ is a variable} \\ A(|\vec{t}|_{w,\nu}) & \text{if } t_a = A(\vec{t}). \end{cases}$$

Note that $\vec{t}$ consists solely of object terms and hence obtains its denotation in the usual way.

The denotation of an arbitrary situation term t_s is defined as follows:

$$|t_s|_{w,\nu} = \begin{cases} \nu(t_s) & \text{if } t_s \text{ is a variable} \\ \langle \omega', \langle \omega' \rangle \rangle & \text{if } t_s = S_0 \text{ and } w = \langle \omega, \omega' | \lambda \rangle. \\ Do(|t_a|_{w,\nu}, |\sigma|_{w,\nu}) & \text{if } t_s = do(t_a, \sigma). \end{cases}$$

Note the way S_0 obtains its denotation. It is always the initial world corresponding to the first element of the action log. This way every world reachable from the same initial world w assigns S_0 the value w.

To define the semantics of formulas of $\mathcal{L}_s$, let $\mathcal{M} = \langle e_I, w_I, Do, poss \rangle$ be an an action model, $w = \langle \omega, \lambda \rangle$ a world, e a set of worlds, and ν a variable map. To say that α is true at e, w, and ν with respect to $\mathcal{M}$, we write $e, w, \nu \models^{\mathcal{M}} \alpha$. We begin with the semantic rules for the formulas in $\mathcal{L}$, which are essentially those from Section 2, except for **O**:

$$\begin{array}{lcl}
e, w, \nu \models^{\mathcal{M}} \mathrm{P}(\vec{t}) & \iff & \vec{n} \in \omega[P] \text{ and } \vec{n} = |\vec{t}|_{w,\nu}, \text{ where } P(\vec{t}) \text{ is atomic and } \vec{t} \text{ are object terms.} \\
e, w, \nu \models^{\mathcal{M}} t_1 = t_2 & \iff & |t_1|_{w,\nu} = |t_2|_{w,\nu}, \text{ where } t_1 \text{ and } t_2 \text{ are obj. terms.} \\
e, w, \nu \models^{\mathcal{M}} \neg\alpha & \iff & e, w, \nu \not\models \alpha \\
e, w, \nu \models^{\mathcal{M}} \alpha \vee \beta & \iff & e, w, \nu \models^{\mathcal{M}} \alpha \text{ or } e, w, \nu \models^{\mathcal{M}} \beta \\
e, w, \nu \models^{\mathcal{M}} \forall x \alpha & \iff & e, w, \nu^x_n \models^{\mathcal{M}} \alpha \text{ for all } n \in \mathcal{N} \\
e, w, \nu \models^{\mathcal{M}} \mathbf{K}\alpha & \iff & \text{for all } w', \text{ if } w' \in e, \text{ then } e, w', \nu \models^{\mathcal{M}} \alpha \\
e, w, \nu \models^{\mathcal{M}} \mathbf{O}\alpha & \iff & \text{for all } w', \text{ if } w' \approx w, \text{ then } w' \in e \text{ iff } e, w', \nu \models^{\mathcal{M}} \alpha
\end{array}$$

Note the change in the definition of **O**. We now only take into account worlds w' that have the same action log as w ($w' \approx w$). To see why this restriction is necessary, consider, for example, an initial state epistemic state e and let p be atomic. When should we say that e only knows p? Clearly, this should be the case when e consists of precisely all *initial* worlds that satisfy p. Any other world, which is not an initial world, yet also satisfies p, should not even be considered. After all, we

are interested in what is known *now*, that is, before any action has taken place. In general, then, given a state (e, w) which resulted from some initial state by performing a sequence of actions λ, **O** should take into account only comparable worlds, that is, worlds that are at the "right" level in (action) history. [11]

What is left to specify is the semantics of the newly introduced primitives and quantification over actions and situations. Let $t_a, t_{a'}$ and $t_s, t_{s'}$ be action and situation terms, respectively.

Equality between actions and situations is defined in the obvious way.

$e, w, \nu \models^{\mathcal{M}} t_a = t_{a'} \iff |\vec{t}_a|_{w,\nu}$ and $|\vec{t}_{a'}|_{w,\nu}$ are identical primitive actions.

$e, w, \nu \models^{\mathcal{M}} t_s = t_{s'} \iff |t_s|_{w,\nu}$ and $|t_{s'}|_{w,\nu}$ are identical worlds.

The predicate *Poss* determines whether an action is possible at a situation. Hence it should receive its meaning from *poss*.

$$e, w, \nu \models^{\mathcal{M}} Poss(t_a, t_s) \iff poss(|t_a|_{w,\nu}, |t_s|_{w,\nu}) = 1.$$

Comp determines whether two situations are comparable.

$$e, w, \nu \models^{\mathcal{M}} Comp(t_s, t_{s'}) \iff |t_s|_{w,\nu} \approx |t_{s'}|_{w,\nu}.$$

Fluents are completely determined by looking up the denotation/truth value of their counterparts in the action model using the situation as the index to find the corresponding world.

$$|f^{fl}(\vec{t}, t_s)|_{w,\nu} = |f(\vec{t})|_{|t_s|_{w,\nu},\nu}$$
$$e, w, \nu \models^{\mathcal{M}} P^{fl}(\vec{t}, t_s) \iff e, |t_s|_{w,\nu}, \nu \models^{\mathcal{M}} P(\vec{t})$$

The predicate $<$ corresponds precisely to the notion of strong reachability.

$e, w, \nu \models^{\mathcal{M}} t_s < t_{s'} \iff |t_{s'}|_{w,\nu}$ is strongly reachable from $|t_s|_{w,\nu}$.

The interpretation of the K-fluent is defined relative to the initial state and all states reachable from there.

$e, w, \nu \models^{\mathcal{M}} K(t_{s'}, t_s) \iff$ for some (e', w') reachable from (e_I, w_I), $|t_{s'}|_{w,\nu} \in e'$ and $|t_{s'}|_{w,\nu} \approx |t_s|_{w,\nu}$.

Let a and s be action and situation variables, respectively. The meaning of $\forall a$ and $\forall s$ is as expected. Note that $\forall^r s$ means reachability relative to S_0, that is, from w's perspective, one looks at all worlds that come before and after w.

[11] The definition of **O** is simpler in the case of *OL* because, in a sense, there are only initial worlds in the model theory to start with.

$e, w, \nu \models^{\mathcal{M}} \forall a\alpha \iff$ for all $A \in \mathcal{A}, e, w, \nu^a_A \models^{\mathcal{M}} \alpha$

$e, w, \nu \models^{\mathcal{M}} \forall s\alpha \iff$ for all $w \in \mathcal{W}, e, w, \nu^s_w \models^{\mathcal{M}} \alpha$

$e, w, \nu \models^{\mathcal{M}} \forall^r s\alpha \iff$ for all w' reachable from $|S_0|_{w,\nu}$, $e, w, \nu^s_{w'} \models^{\mathcal{M}} \alpha$

This concludes the discussion of the semantic rules and all that is left is the definition of validity, which is the usual. A formula α of $\mathcal{L}_s$ is **valid** ($\models \alpha$) iff for all action models $\mathcal{M} = \langle e, w, Do, poss\rangle$ and for all variable maps ν, $e, w, \nu \models^{\mathcal{M}} \alpha$.

In the rest of this section we will discuss various properties of the logic.

OL as part of *OLS*

As one would expect, when restricted to the language $\mathcal{L}$, *OLS* coincides with *OL*.

Theorem 1 *For any $\alpha \in \mathcal{L}$, $\models_{OL} \alpha$ iff $\models \alpha$.*

Proof : The result follows from the fact that every model of α in *OL* can easily be mapped into a model of α in *OLS* and vice versa. Let $\alpha \in \mathcal{L}$.

1. For any valuation w, set of valuations e, and variable map ν, $e, w, \nu \models \alpha$ iff $e\uparrow, w\uparrow, \nu \models^{\mathcal{M}} \alpha$, where $\mathcal{M} = \langle e\uparrow, w\uparrow, Do, poss\rangle$ for arbitrary *Do* and *poss*.
2. For any action model $\mathcal{M} = \langle e, w, Do, poss\rangle$ and variable map ν, $e, w, \nu \models^{\mathcal{M}} \alpha$ iff $e\downarrow, w\downarrow, \nu \models \alpha$.

The proof proceeds by a straightforward induction on the structure of α (omitted). From this the theorem follows immediately. ∎

The proof of the theorem demonstrates that, given an action model $\mathcal{M} = \langle e, w, Do, poss\rangle$, valid formulas in $\mathcal{L}$ are only about the initial states of action models. In this case, it is completely immaterial what the action execution and precondition functions are.

Lemma 4.1 *Let $\mathcal{M} = \langle e, w, Do, poss\rangle$ and $\mathcal{M}' = \langle e, w, Do', poss'\rangle$, that is, $\mathcal{M}$ and $\mathcal{M}'$ have the same initial state and differ only in the action execution and precondition functions. Then for any $\alpha \in \mathcal{L}$ and variable map ν, $e, w, \nu \models^{\mathcal{M}} \alpha$ iff $e, w, \nu \models^{\mathcal{M}'} \alpha$.*

Similarly, any two states reachable from the initial state that differ only in the action logs but not in the valuations satisfy precisely the same formulas in $\mathcal{L}$.

Lemma 4.2 *Let $\mathcal{M} = \langle e, w, Do, poss\rangle$. Let (e_1, w_1) and (e_2, w_2) be two states reachable from (e, w) such that $e_1\downarrow = e_2\downarrow$ and $w_1\downarrow = w_2\downarrow$. Then for any $\alpha \in \mathcal{L}$ and variable map ν, $e_1, w_1, \nu \models \alpha$ iff $e_2, w_2, \nu \models \alpha$.*

While perhaps reassuring, the fact that we have recreated *OL* as part of *OLS* is neither surprising nor terribly interesting in its own right. The main purpose

of OLS, after all, is to be able to investigate states other than initial states, where we need the language of the situation calculus to do so. In this respect, having OL as part of OLS turns out to be useful from a technical point of view when we prove our main result about a strong correspondence between properties of OL and properties of arbitrary states reachable from initial states.

However, before going into that, let us turn to the foundational axioms that are associated with the usual proof theoretic accounts of the situation calculus and show that these are valid in OLS.

Foundational axioms of the situation calculus

Here we consider the foundational axioms as proposed, for exam ple, in [LR94].

Theorem 2 *The following sentences are valid in* OLS.

1. $\forall s \forall a(S_0 \neq do(a,s))$.
2. $\forall s \neg(s < S_0)$.
3. $\forall a_1 \forall a_2 \forall s_1 \forall s_2 [do(a_1,s_1) = do(a_2,s_2) \supset (a_1 = a_2 \wedge s_1 = s_2)]$.
4. *Let* $s \leq s'$ *be an abbreviation for* $s < s' \vee s = s'$. *Then* $\forall s(s < do(a,s') \equiv (Poss(a,s') \wedge s \leq s')$.
5. *The following 2nd order sentence is the equivalent of Reiter's induction axiom for situations quantifying over predicates* P. *Note that* s *is restricted to reachable worlds only.*

   $\forall P[P(S_0) \wedge [\forall^r s \forall a(P(s) \supset P(do(a,s)))] \supset \forall^r s P(s)]$

Proof:

1. To prove that $\models \forall s \forall a(S_0 \neq do(a,s))$, assume, to the contrary, that there is an action model $\mathcal{M} = \langle e, w, Do, poss \rangle$, a world $w' \in \mathcal{W}$ and a primitive action $A \in \mathcal{A}$ such that $e, w, \nu^{s,a}_{w',A} \models^{\mathcal{M}} S_0 = do(a,s)$. We know that $|S_0|_{w,\nu} = w = \langle \omega, \langle \omega \rangle \rangle$ for some valuation ω. Let $w' = \langle \omega', \lambda \rangle$ for some action log λ. Then $|do(a,s)|_{w,\nu} = w^* = \langle \omega^*, \lambda | A \rangle$. Hence, no matter what the effect of A is, the action log of w^* differs from that of w. Hence the two cannot be the same worlds.
2. Assume, to the contrary, that $e, w, \nu^{s}_{w'} \models^{\mathcal{M}} s < S_0$ for some $\mathcal{M} = \langle e, w, Do, poss \rangle$ and $w' \in \mathcal{W}$. Thus, by the definition of $<$, $|S_0|_{w,\nu} = w$ is strongly reachable from w', that is, $w = Do(A, w^*)$ for some w^*, contradicting (1.).
3. To prove the validity of the unique names axiom for actions, let $\mathcal{M} = \langle e, w, Do, poss \rangle$ and ν a variable map. Let $\nu' = \nu^{a_1\, a_2\, s_1\, s_2}_{A_1\, A_2\, w_1\, w_2}$ for actions A_1 and A_2 and worlds w_1 and w_2 and let $w'_1 = Do(\nu'(a_1), \nu'(s_1))$ and $w'_2 = Do(\nu'(a_2), \nu'(s_2))$. Now let $e, w, \nu' \models^{\mathcal{M}} do(a_1,s_1) = do(a_2,s_2)$, that is, $w'_1 = w'_2$. Thus w'_1 and w'_2 agree on their action logs and, in particular, on the last action in the log, which implies that $\nu'(a_1) = \nu'(a_2)$. Also, since w'_1 and w'_2 agree on the entire action log, they both resulted from the same initial world by the same actions executed in the same order. Since actions are deterministic, all intermediate worlds starting from the initial world are identical. Hence $\nu'(s_1) = \nu'(s_2)$.
4. Follows immediately from the definition of strong reachability.
5. The validity of the induction axiom follows by a simple induction over the worlds reachable from the denotation of S_0. ∎

Does the induction axiom hold if we replace $\forall^r$ by the unrestricted $\forall$? The answer is no. For example, let $\mathcal{M} = \langle e, w, Do, poss_0 \rangle$ be an action model such that no actions are possible, that is $poss_0(A,w) = 0$ for all A and w. Let P be any property which holds at S_0. Then P holds trivially at every reachable world. However, if P is falsifiable, then there exists an initial world where P is false. Also, induction makes no sense in the first place when talking about all situations, meaning all worlds, since there are uncountably many of them. Reiter [Rei95] already noticed that the K-fluent bodes ill for the induction axiom since it is incompatible with epistemic alternatives to S_0 which cannot be reached by an action. Here we have a clear picture: while induction must be given up as far as all situations are concerned, it can be salvaged when restricted to reachable situations.

Properties of arbitrary reachable states

While the situation calculus and, thereby, OLS allow us to form statements that mention an arbitrary number of different situations, this full generality is hardly ever used. In fact, often one wants to represent and reason about what holds at one particular situation. For example, when planning in the situation calculus, one may need to reason about what is true or what is known after performing an action in a given state in order to pick the next appropriate action. In the situation calculus, a principled way to talk about a particular situation s is to take a first-order formula α that does not mention situations and translate it into a formula $\alpha[s]$ which expresses in an appropriate way using fluents involving s that α holds at s. A simple example is an atomic formula $P(a)$, where $P(a)[s] = P^{fl}(a,s)$. Here we generalize this approach and consider arbitrary formulas of $\mathcal{L}$ including those involving the modal operators $\mathbf{K}$ and $\mathbf{O}$. The main result will be that a formula of the form $\alpha[s]$ is valid in OLS precisely when α is valid in OL. The main benefit is that all the results about OL, in particular,

reasoning about only knowing, can be imported into the situation calculus.

First, however, we turn to the translation from a formula α in $\mathcal{L}$ into a situation calculus formula involving a situation s. The translation proceeds, roughly, by replacing every term and atomic formula by its fluent counterpart and replacing the operators **K** and **O** by expressions involving the K-fluent. As for **K**, the translation is the usual situation calculus way of expressing knowledge. For example, $\mathbf{K}P(\#1)[s]$ turns into $\forall s' K(s', s) \supset P^{fl}(\#1, s')$. Similarly, and in a way completely analogous to the semantic rule for **O** in OLS, $\mathbf{O}P(\#1)[s]$ translates into $\forall s' Comp(s', s) \supset [K(s', s) \equiv P^{fl}(\#1, s')]$. In general:

Definition 8 *Given any term or formula ϕ in $\mathcal{L}$, the corresponding term or formula $\phi[s]$ in $\mathcal{L}_s$, where s is any situation term, is defined as follows.*

$x[s] = x$ *if* x *is a variable*
$n[s] = n$ *if* n *is a standard name*
$f(t_1, \ldots, t_n)[s] = f^{fl}(t_1, \ldots, t_n, s)$
  if $f(\vec{t})$ *is a term in* $\mathcal{L}$
$P(t_1, \ldots, t_n)[s] = P^{fl}(t_1, \ldots, t_n, s)$
  if $P(\vec{t})$ *is an atomic formula in* $\mathcal{L}$
$(t_1 = t_2)[s] = (t_1[s] = t_2[s])$
$(\neg\alpha)[s] = \neg\alpha[s]$
$(\alpha \vee \beta)[s] = \alpha[s] \vee \beta[s]$
$(\exists x \alpha) = \exists x \alpha[s]$
$(\mathbf{K}\alpha)[s] = \forall s' K(s', s) \supset \alpha[s']$
$(\mathbf{O}\alpha)[s] = \forall s' Comp(s', s) \supset (K(s', s) \equiv \alpha[s'])$

Before we get to the main result relating the validity of α in OL to the validity of $\alpha[s]$ in OLS, we need the following lemmas.

Lemma 4.3 *Let $\mathcal{M} = \langle e, w, Do, poss\rangle$, (e', w') a state reachable from (e, w), ν a variable map, s a situation term, and w'' a world such that $w'' = |s|_{w,\nu}$ and $w'' \approx w'$. Then for any object term t and any $\alpha \in \mathcal{L}$,*

1. $|t[s]|_{w,\nu} = |t|_{w'',\nu}$
2. $e, w, \nu \models^{\mathcal{M}} \alpha[s]$ iff $e', w'', \nu \models^{\mathcal{M}} \alpha$.

Proof:

1. There are three cases to consider:

a) $t = x$ (variable).
$|x[s]|_{w,\nu} = |x|_{w,\nu} = \nu(x) = |x|_{w'',\nu}$.

b) $t = n$ (standard name).
$|n[s]|_{w,\nu} = |n|_{w,\nu} = n = |n|_{w'',\nu}$.

c) $t = f(\vec{u})$.
$|f(\vec{u})[s]|_{w,\nu} = |f^{fl}(\vec{u}, s)|_{w,\nu} = |f(\vec{u})|_{w'',\nu}$.

2. By induction on the structure of α. For an atomic formula $P(\vec{t})$ we have $e, w, \nu \models^{\mathcal{M}} P(\vec{t})[s]$ iff $e, w, \nu \models^{\mathcal{M}} P^{fl}(\vec{t}, s)$ iff $e, |s|_{w,\nu}, \nu \models^{\mathcal{M}} P(\vec{t})$ (by definition) iff $e, w'', \nu \models^{\mathcal{M}} P(\vec{t})$ iff $e', w'', \nu \models^{\mathcal{M}} P(\vec{t})$ (since the epistemic state plays no role here).

In the case of an atomic formula $u = v$ we get $e, w, \nu \models^{\mathcal{M}} (u = v)[s]$ iff $e, w, \nu \models^{\mathcal{M}} u[s] = t[s]$ iff $|u|_{w'',\nu} = |v|_{w'',\nu}$ (by (1.)) iff $e', w'', \nu \models^{\mathcal{M}} u = v$.

The cases for $\neg$, $\vee$, and $\forall$ follow immediately by induction.

$e, w, \nu \models^{\mathcal{M}} \mathbf{K}\alpha[s]$ iff $e, w, \nu \models^{\mathcal{M}} \forall s' K(s', s) \supset \alpha[s']$ iff $\forall w^* \in \mathcal{W}, e, w, \nu^{s'}_{w^*} \models^{\mathcal{M}} K(s', s) \supset \alpha[s']$ iff $\forall w^* \in \mathcal{W}$, if $w^* \in e'$, then $e, w, \nu^{s'}_{w^*} \models^{\mathcal{M}} \alpha[s']$ iff $\forall w^* \in e', e', w^*, \nu^{s'}_{w^*} \models^{\mathcal{M}} \alpha$ (by induction, since $|s'|_{w,\nu} = w^* \approx w'$) iff $\forall w^* \in e', e', w^*, \nu \models^{\mathcal{M}} \alpha$ (since α mentions no situations) iff $e', w, \nu \models^{\mathcal{M}} \mathbf{K}\alpha$.

$e, w, \nu \models^{\mathcal{M}} \mathbf{O}\alpha[s]$
iff $e, w, \nu \models^{\mathcal{M}} \forall s' Comp(s', s) \supset (K(s', s) \equiv \alpha[s'])$ iff $\forall w^* \in \mathcal{W}$, if $w^* \approx |s|_{w,\nu}$ then $e, w, \nu^{s'}_{w^*} \models^{\mathcal{M}} K(s', s) \equiv \alpha[s']$ iff $\forall w^* \approx w, w^* \in e'$ iff $e, w, \nu^{s'}_{w^*} \models^{\mathcal{M}} \alpha[s']$ iff $\forall w^* \approx w, w^* \in e'$ iff $e', w^*, \nu \models^{\mathcal{M}} \alpha$ (by induction) iff $e', w, \nu \models^{\mathcal{M}} \mathbf{O}\alpha$. ∎

Lemma 4.4 *Let $\mathcal{M} = \langle e, w, Do, poss_0\rangle$, where $poss_0(A, w) = 0$ for all actions A and worlds w. In other words, no action is possible at any world. Then for any state (e', w') reachable from (e, w), $e\downarrow = e'\downarrow$ and $w\downarrow = w'\downarrow$.*

Proof: By induction on the number of actions needed to reach (e', w'). If no actions are necessary, the lemma holds trivially since $e = e'$ and $w = w'$. Assume the lemma holds for states reachable in n actions. Let (e', w') be reachable in $n + 1$ actions. The $e' = Do^e(A, e'', w'')$ and $w' = Do(A, w'')$, where (e'', w'') is reachable in n actions. Since $poss_0(A, w) = 0$ for all w, we immediately get from the definitions of Do and Do^e that $w'\downarrow = w''\downarrow$ and $e'\downarrow = e''\downarrow$. Finally $w''\downarrow = w$ and $e''\downarrow = e$ follows by induction. ∎

Theorem 3 *For any formula $\alpha \in \mathcal{L}$ and closed situation term s, $\models \alpha$ iff $\models \alpha[s]$.*

Proof: To prove the "only if" direction, assume $\models \alpha$ and let $\mathcal{M} = \langle e, w, Do, poss\rangle$ be an action model. We need to show that $e, w, \nu \models^{\mathcal{M}} \alpha[s]$ for an arbitrary variable map ν. Let $w^* = |s|_{w,\nu}$ and (e', w') a state reachable from (e, w) such that $w' \approx w^*$. (Such (e', w') obviously exists.) Since $\models \alpha$ and with the help of Lemmas 4.1, 4.2, and 4.4, it is not hard to show that $e', w', \nu \models^{\mathcal{M}} \alpha$. Hence, by Lemma 4.3, $e, w, \nu \models \alpha[s]$.

For the converse, assume $\models \alpha[s]$. Let $\mathcal{M} = \langle e, w, Do, poss\rangle$. We need to show that $e, w, \nu \models^{\mathcal{M}} \alpha$ for an arbitrary ν. Consider $\mathcal{M}' = \langle e, w, Do, poss_0\rangle$ with $poss_0(A, w^*) = 0$ for all A and w^* and let $w' = |s|_{w,\nu}$. Since s is a closed term, w' is reachable from the denotation of S_0, which is w. In particular, (e', w') is

reachable from (e, w) for some e' in $\mathcal{M}'$. (Note that reachability is independent of actions being possible or not.) Since $\models \alpha[s]$, we have $e, w, \nu \models^{\mathcal{M}'} \alpha[s]$. Then, by Lemma 4.3, $e', w', \nu \models^{\mathcal{M}'} \alpha$ and, by Lemma 4.4, $e'\downarrow = e\downarrow$ and $w'\downarrow = w\downarrow$. Hence, by Lemma 4.2, $e, w, \nu \models^{\mathcal{M}'} \alpha$ and finally, by Lemma 4.1, $e, w, \nu \models^{\mathcal{M}} \alpha$. ∎

Together with Theorem 1 we immediately obtain the following:

Corollary 4.5 *For any formula $\alpha \in \mathcal{L}$ and closed situation term s, $\models_{OL} \alpha$ iff $\models \alpha[s]$.*

As said earlier, this result allows us to import everything we know about *OL* when analyzing statements of the form $\alpha[s]$. For example, we can infer immediately that knowledge at situations reachable from S_0 has all the properties of *S5*. Perhaps more importantly, it allows us to use results of *OL* to determine what an agent knows about the world in situation s given that he or she only knows KB[s], where KB[s] should be thought of as the agent's knowledge base at s. In particular, Levesque has shown that if KB is objective, then determining whether an arbitrary sentence is known reduces to first-order reasoning alone, that is, no modal machinery is necessary or, equivalently, we do not have to deal at all with the K-fluent, which is reassuring [Lev90].[12]

We end this discussion with a brief example demonstrating that *OLS* models the kind of reasoning necessary to deal with scenarios like those outlined in the introduction.

Example 4.1
Consider the sentence $\alpha = \forall x Room(x) \equiv (x = B)$, where B is not a standard name, and suppose this is all the agent knows about the world in the initial situation S_0. From Theorem 3 and the properties of OL we obtain:

$$\models (\mathbf{O}\alpha)[S_0] \supset (\mathbf{K}\exists x Room(x) \wedge \neg \mathbf{K} Room(x))[S_0]$$

Let A^B be the sense action that determines the standard name of room B and assume B refers to #18 in the real world. After doing the sensing the agent should not only know that $B = \#18$ but also that it now knows all the rooms, since there is only one. This is indeed what the logic predicts:

$$\models (B = \#18) \wedge Poss(A^B, S_0) \wedge (\mathbf{O}\alpha)[S_0] \supset (\mathbf{K}(B = \#18))[do(A^B, S_0)].$$
$$\models (B = \#18) \wedge Poss(A^B, S_0) \wedge (\mathbf{O}\alpha)[S_0] \supset (\mathbf{K}\forall x Room(x) \supset \mathbf{K} Room(x))[do(A^B, S_0)].$$

[12] Note that, while our version of *OL* is slightly different from Levesque's, these particular results carry over without modification.

Knowledge and Action

In this final subsection we analyze in more detail the interaction between knowledge and action in *OLS*. In particular, we show that Moore's axioms [Moo85a][13] which determine how the K-fluent changes in response to physical and sense actions are valid in *OLS*.

We begin with the corresponding result for physical actions, which states that the situations epistemically accessible after performing the action are exactly those situations that result from performing the action in those situations that were accessible before the action was executed.

Theorem 4 *Let A be a primitive physical action. Then*

$$\models \forall s_1 \forall s_2 K(s_2, do(A, s_1)) \equiv [\exists s_3 K(s_3, s_1) \wedge s_2 = do(A, s_3)].$$

Proof : Let $AM = \langle e, w, Do, poss \rangle$ be an action model. It suffices to show that for any worlds w_1, w_2, and variable map ν, $e, w, \nu^{s_1 s_2}_{w_1 w_2} \models^{\mathcal{M}} K(s_2, do(A, s_1))$ iff $e, w, \nu^{s_1 s_2}_{w_1 w_2} \models^{\mathcal{M}} \exists s_3 K(s_3, s_1) \wedge s_2 = do(A, s_3)$.

To prove the "only if" direction, let $e, w, \nu^{s_1 s_2}_{w_1 w_2} \models^{\mathcal{M}} K(s_2, do(A, s_1))$. Then there is an e_2 reachable from (e, w) such that $w_2 \in e_2$ and $w_2 \approx |do(A, s_1)|_{w,\nu} = Do(A, w_1)$. This implies that $w_2 = Do(A, w_3)$ for some $w_3 \in e_3$, where e_3 is reachable from (e, w). Since $w_2 \approx Do(A, w_1)$, we also have $w_1 \approx w_3$. Thus there is an e_3 reachable from (e, w) and a $w_3 \in e_3$ such that $w_3 \approx w_1$ and $w_2 = Do(A, w_3)$, that is, $e, w, \nu^{s_1 s_2}_{w_1 w_2} \models^{\mathcal{M}} \exists s_3 K(s_3, s_1) \wedge s_2 = do(A, s_3)$.

Conversely, let $e, w, \nu^{s_1 s_2}_{w_1 w_2} \models^{\mathcal{M}} \exists s_3 K(s_3, s_1) \wedge s_2 = do(A, s_3)$. Then there is an e_3 reachable from (e, w) and a $w_3 \in e_3$ such that $w_3 \approx w_1$ and $w_2 = Do(A, w_3)$. Thus there is an e_2 reachable from (e, w) with $w_2 \in e_2$. Since $w_2 = Do(A, w_3)$ and $w_1 \approx w_3$, we also have $w_2 \approx Do(A, w_1)$. Hence, $e, w, \nu^{s_1 s_2}_{w_1 w_2} \models^{\mathcal{M}} K(s_2, do(A, s_1))$. ∎

Note that the equivalence holds independent of whether A is possible or not. This is because we defined $Do(A, w)$ in such a way that it behaves like a noop in case A is impossible to execute.

As for sense actions, we obtain the following:

Theorem 5 *Let A^ϕ and A^τ be sense actions, where ϕ is an objective sentence and τ a closed term in $\mathcal{L}$. Then*

1. $\models \forall^\tau s_1 \forall s_2 Poss(A^\phi, s_1) \supset K(s_2, do(A^\phi, s_1)) \equiv (\exists s_3 K(s_3, s_1) \wedge s_2 = do(A^\phi, s_3) \wedge \phi[s_2] \equiv \phi[do(A^\phi, s_1)])$.
2. $\models \forall^\tau s_1 \forall s_2 Poss(A^\tau, s_1) \supset K(s_2, do(A^\tau, s_1)) \equiv (\exists s_3 K(s_3, s_1) \wedge s_2 = do(A^\tau, s_3) \wedge (\tau[s_2] = \tau[do(A^\phi, s_1)]))$.

[13] Here we use the notation of [SL93].

Proof: Here we only prove the case for sensing the value of an objective sentence ϕ. The case of sensing the referent of a term is handled analogously.

Let $\mathcal{M} = \langle e, w, Do, poss \rangle$. Let (e_1, w_1) be an arbitrary state reachable from (e, w) and w_2 any world. Let $e, w, \nu^{s_1 s_2}_{w_1 w_2} \models^{\mathcal{M}} Poss(A^\phi, s_1)$, that is, $poss(A^\phi, w_1) = 1$. Let (e', w') be the state resulting from (e_1, w_1) by performing A^ϕ, that is, $e' = Do^e(A^\phi, e_1, w_1) = \{w^* \mid \exists w^{**} \in e_1, w^* = Do(A^\phi, w^{**})$ and $e\downarrow, w^{**}\downarrow \models \phi$ iff $e\downarrow, w\downarrow \models \phi.\}$ and $w' = Do(A^\phi, w_1)$. Then $e, w, \nu^{s_1 s_2}_{w_1 w_2} \models^{\mathcal{M}} K(s_2, do(A^\phi, s_1))$ iff $w_2 \in e'$ iff $w_2 = Do(A^\phi, w_3)$ for some $w_3 \in e_1$ and $(e_1, w_3, \nu^{s_1 s_2}_{w_1 w_2} \models^{\mathcal{M}} \phi$ iff $e_1, w_1, \nu^{s_1 s_2}_{w_1 w_2} \models \phi)$ iff $w_2 = Do(A^\phi, w_3)$ for some $w_3 \in e_1$ and $(e, w, \nu^{s_1 s_2 s_3}_{w_1 w_2 w_3} \models^{\mathcal{M}} \phi[s_3]$ iff $e, w, \nu^{s_1 s_2 s_3}_{w_1 w_2 w_3} \models \phi[s_1])$ iff $w_2 = Do(A^\phi, w_3)$ for some $w_3 \in e_1$ and $(e, w, \nu^{s_1 s_2 s_3}_{w_1 w_2 w_3} \models^{\mathcal{M}} \phi[s_2]$ iff $e, w, \nu^{s_1 s_2 s_3}_{w_1 w_2 w_3} \models \phi[do(A^\phi, s_1)])$
[since w_3 and w_1 agree in their valuations with w_2 and $Do(A^\phi, w_1)$, respectively, and since ϕ is objective] iff $e, w, \nu^{s_1 s_2}_{w_1 w_2} \models^{\mathcal{M}} \exists s_3 K(s_3, s_1) \wedge s_2 = do(A^\phi, s_3) \wedge \phi[s_2] \equiv \phi[do(A^\phi, s_1)]$. ∎

The intuitive reading of (1.) is that after sensing the truth value of ϕ in a situation reachable from S_0, every epistemically accessible situation will have the same truth value for ϕ as the current (actual) situation. In other words, the truth value of ϕ becomes known.

Note that the theorem has two important restrictions. For one, we only consider situations s_1 that are reachable from S_0. For another, the sentence whose truth value is being sensed must be objective. The first restriction is necessary because sensing is done always with respect to a world which could qualify as the real world. For a given action model $\mathcal{M} = \langle e, w, Do, poss \rangle$, these are only the worlds reachable from w. The second restriction is a little more subtle. Let $\phi = p \wedge \neg \mathbf{K} p$ for some atomic p and let $\mathcal{M} = \langle e, w, Do, poss \rangle$ such that p is not known in the initial state (e, w), but p is true in the initial real world (w), that is, $e, w, \nu \models^{\mathcal{M}} \phi[S_0]$. Ignoring the fact that actions update the action log of worlds, sensing ϕ has essentially the effect of removing from e all the worlds that falsify p. The resulting state $(e', w') = (Do^e(A^\phi, e, w), Do(A^\phi, w))$, however, does not know ϕ since p is now known to be true. In particular, $e, w, \nu \models^{\mathcal{M}} \neg\phi[do(A^\phi, S_0)]$. And this is how it should be since what we know about our knowledge about the world changes nonmonotonically when we gain more information about the world. Finally, in contrast to Theorem 4, the assumption that the sense action is possible makes a difference now. In fact, if A^ϕ (and similarly for A^τ) is not possible, we get essentially a version of Theorem 4, that is, $\forall s_1 \forall s_2 \neg Poss(A^\phi, s_1) \supset K(s_2, do(A, s_1)) \equiv \exists s_3 K(s_3, s_1) \wedge s_2 = do(A, s_3)$ is valid. This is because we have chosen our model of action in such a way that an impossible sense action acts like a (physical) noop action.

5 Conclusions

In this paper, we integrated Levesque's logic of only knowing into the situation calculus, thereby providing a richer framework for reasoning about knowledge and action. In doing so, it was necessary to reconstruct the situation calculus by developing a model theory of action based on possible-world semantics and explicitly interpreting situations as possible worlds.

Much more needs to be done, though. It seems that we have only scratched the surface in analyzing *OLS* and realizing the full potential of this approach. While a complete axiomatization may be far off given the problems already encountered in *OL*, the approach offers other interesting avenues of investigation. For example, Lin and Reiter [LR94] define a notion of database progression in the situation calculus, which is sometimes not first-order definable. In a forthcoming paper [Lak9x], we show that our model theory of the situation calculus allows us to define a notion of database progression which is always first-order definable. Also, we have not even touched upon the fact that autoepistemic logic [Moo85b] is completely characterizable in terms of only knowing, as shown in [Lev90]. Hence we obtain a way to integrate nonmonotonic reasoning with a theory of action. Actually, using belief rather than knowledge would seem more appropriate in this context. Again, it is not hard to modify our framework to deal with belief.

References

[BHL95] Bacchus, F., Halpern, J. Y., and Levesque, H. J., Reasoning about Noisy Sensors in the Situation Calculus. Draft. A short version of the paper appeard in: *Proc. of the 13th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, 1995, pp. 1933–1940.

[BBC95] Buhmann, J., Burgard, W., Cremers, A. B., Fox, D., Hofmann, T., Schneider, F., Strikos, J, and Thrun, S., The Mobile Robot RHINO. *AI Magazine* bf 16(2), 1995, pp. 31–38.

[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.

[Kri63] Kripke, S. A., Semantical considerations on modal logic. *Acta Philosophica Fennica* **16**, 1963, pp. 83–94.

[Lak9x] Lakemeyer, G., First-Order Database Progression, 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.

[LR94] Reiter, R. and Lin, F., How to Progress a Database (and Why) I: Formal Foundations. In *Proc. Fourth Int. Conf. on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, 1994, pp. 425-436.

[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.

[Mor88] Moravec, H. P., Sensor Fusion in Certainty Grids for Mobile Robots, *AI Magazine*, 1988, pp. 61–74.

[Pin94] Pinto, J. A., *Temporal Reasoning in the Situation Calculus*, Ph.D. Thesis, Department of Computer Science, University of Toronto, 1994.

[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.

[Rei95] Reiter, R., Course Notes for CS2532, University of Toronto, Dept of Computer Science, 1995.

[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.

Modeling Complex Systems in the Situation Calculus: A Case Study Using the Dagstuhl Steam Boiler Problem

Todd G. Kelley
Department of Computer Science
University of Toronto
Toronto, Canada M5S 1A4
email: tgk@cs.toronto.edu

Abstract

The situation calculus is showing tremendous promise as a formal framework for modeling the dynamical worlds encountered in real life. We take advantage of the results of [Rei96] to show that the situation calculus is a powerful and practical modeling language. The paper provides a brief overview of the concurrent temporal situation calculus and how it is used to specify physical behavior. It then presents the Dagstuhl steam boiler problem as an example of a complex physical system of interest in the real world. The problem was the focus of the Dagstuhl meeting, "Methods for Semantics and Specification", whose goal was to develop criteria by which to compare advantages and drawbacks of formal methods for practical applications. The paper presents the situation calculus specification of the focus of the problem, the steam boiler controller. It then discusses the theoretical foundation of a PROLOG technology simulator, which, together with the specification, form an implementation of the controller. The paper concludes with an evaluation of the situation calculus solution to the problem, using the criteria which emanated from the Dagstuhl meeting.

1 Introduction

The situation calculus language ([MH69]) has received much attention from the Cognitive Robotics Group at the University of Toronto in recent years. The language is showing tremendous promise as a formal framework for modeling the dynamical worlds encountered in real life.

The challenges facing such a framework are numerous. For example, it must facilitate the representation of time, continuous processes, actions performed by agents with "free will", actions performed by Nature, non-deterministic actions of chance, knowledge-producing actions and the mental state of agents, concurrent or simultaneous actions, etc. The framework must also be conducive to various types of reasoning, including prediction, planning, diagnosis, and hypothetical reasoning. A single formal theory of action and time that satisfies these conditions is the ongoing objective of the Cognitive Robotics Group.

Encouraging progress towards this long-range objective using the situation calculus as the framework has been achieved [Rei91, Pin94, LRL+96, LLL+96, SL93]. This paper concentrates specifically on the results of [Rei96] which make it possible to formally model the behavior of physical systems as complex as a steam boiler controller. These physical systems (the toilet of [Kel96] is another example) involve time, continuous processes, and simultaneous natural actions (those dictated by the laws of physics). The situation calculus of [Rei96] provides a knowledge representation framework that is conducive to the specification and simulation of such systems, while explicitly embodying a solution to the frame and qualification problems.

As a specification language, the situation calculus boasts many desirable properties. For example, a situation calculus model of a physical system is a truly logical specification of that system. Hence, items of interest, such as behaviors of parameters, are *logical consequences* of the specification. This feature of the situation calculus is clearly conducive to formal verification. Furthermore, the foundational axioms of the situation calculus provide a firm theoretical foundation for a situation calculus-based PROLOG simulator of situation calculus specifications. The behavioral properties of a situation calculus specification can be obtained automatically by directly executing the specification on the simulator. *Hence, a specification in situation calculus form is also an implementation whose behavioral properties are automatically formally verified against the specification.*

To illustrate that the situation calculus is a practical modeling language, we have formalized the controller

specification for the Dagstuhl steam boiler [Abr94]. The original text from which the specification is derived was written by LtCol. J. C. Bauer for the Institute for Risk Research of the University of Waterloo, and submitted as a competition problem to be solved by the participants of the International Software Safety Symposium organized by the Institute for Risk Research. The Dagstuhl steam boiler problem, solved in this paper, stems from that original text. The problem was the focus of the Dagstuhl meeting, "Methods for Semantics and Specification", whose goal was to develop criteria by which to compare advantages and drawbacks of formal methods for practical applications. Hence, the problem is ideal for exhibiting the features of the situation calculus. In this paper, I present an evaluation of the situation calculus solution to the problem, using the criteria which emanated from the Dagstuhl meeting [ABL95].

2 Situation Calculus Ontology

The situation calculus is designed to formalize the behavior of dynamically changing worlds. Intuitively there are two facets to the ontology itself: 1) distinguishing between different courses of action, and 2) determining the state of the world after different courses of action. There are two additional facets of the situation calculus: 3) axioms which specify which courses of action can happen, and 4) axioms which specify the results of courses of action.

2.1 Naming Courses of Action

The mechanism for all change in such worlds is one or more agents, perhaps including Nature, performing named, instantaneous, *actions*. *Situations* are histories of concurrent action occurrences, each denoting a different possible evolution of the world. A concurrent action is a set of simultaneous simple actions. The constant symbol S_0 denotes the initial situation in which no actions have yet occurred. Other than S_0, all situations have names of the form, $do(\alpha, \sigma)$, intuitively meaning the result of doing action α in situation σ. Actions are denoted by functions, with time t being the last parameter. For example, consider the situation

$$do(\{stop\_talking(T_2)\},\\ do(\{begin\_walking(T_1), begin\_talking(T_1)\}, S_0),$$

which denotes the world history in which an agent begins walking and talking at time T_1, and then stops talking at time T_2. The agent would not be talking in this situation, but she would still be walking. The first concurrent action performed is $\{begin\_walking(T_1), begin\_talking(T_1)\}$, which is the set consisting of two simple actions $begin\_walking(T_1)$, and $begin\_talking(T_1)$. Given any situation, the order of action occurrences is obtained by scanning the situation from right to left. So, this situation denotes a world history corresponding to the action sequence

$$[\{begin\_walking(T_1), begin\_talking(T_1)\},\\ \{stop\_talking(T_2)\}].$$

2.2 The State of the World

The state of a world resulting from a certain course of action is determined by the values of *fluents*.

Relational fluents are denoted by predicate symbols taking a situation term as their last argument. These relations represent what is true about the world after carrying out the course of action specified by their situation argument. For example, the fluent, $happy(p, s)$ might mean that person p is happy in s. Note that technically a situation is not a state, but a history of action occurrences; so in this context, we should take "p is true *in* s" to mean, "p is true *after carrying out, in order, all and only the actions specified by* s".

Functional fluents are functions whose value varies from one situation to another. For example, $const\_position(ball, s)$ might denote the real-valued constant position of a ball in situation s.

Continuous processes are represented using functional fluents. The important idea, due to Pinto [Pin94], is that although a continuous process involves continuous change in the values of one or more parameters, the values of the parameters can be modeled by equations which do not change in a particular situation. We say that the *behavior* of the parameter is constant. Unlike the QSIM-style qualitative reasoners (e.g. [Kui86]), which abstract change over points and intervals from the continous change of a parameter, a situation calculus treatment abstracts behavior from the continuous change. This treatment of continuous processes is in the spirit of Sandewall's work [San89]. This work introduces the idea of using differential equations to represent the behavior of parameters, and using logic to specify which differential equations model the behavior of a parameter during the different phases of the evolution of a dynamical system. In contrast to the situation calculus, Sandewall's logic is non-monotonic.

Consider a ball's position, which varies with time in s. The $position(ball, s)$ fluent has as its value a function of time. Also consider the function, $val(f, t)$, which takes as arguments a function of time, f, and a time, t, and whose value is the value of f at t. We take $f(t)$ to be an abbreviation for $val(f, t)$, and we could write[1] $position(ball, s)(t) = x_0 - 1/2gt^2$, meaning that the value of the position function of the ball at time t in situation s is $x_0 - 1/2gt^2$. The position function in a certain situation might be defined

[1] In this paper, lower case Roman characters denote variables. Also, free variables are implicitly universally prenex quantified.

on the entire real line, as it is in this case, but it is only relevant to the model on some half-open interval: $[start(s), \infty)$, or $[start(s), start(do(c, s)))$ for some c, where $start(do(c, s))$ is defined to be $time(c)$, and $time(c)$ denotes the time at which c occurs.

In the situation calculus, all change must be the result of some action occurrence, and functional fluents are consistent with this. In the case of the ball, although the ball's position varies in s, its position follows a single function in s. The behavior represented by the $position(x, s)$ fluent remains unchanged until some action (perhaps $catch(x, t)$) takes place to change it.

2.3 Specifying Which Courses of Action Can Happen

Precondition axioms determine the conditions under which an action is possible. A formalization of a world includes one precondition axiom for each action. For example, the precondition axiom for the $bounce(ball, t)$ action states that the ball bounces if it is falling and it is at the floor.

Consider the function, $\frac{d}{dt}(f)$, which takes a function of time, f, as its argument and has as its value the function which is the time rate of change of f. We write $\frac{d}{dt}(position(ball, s))(t) < 0$ to say that the value of the time rate of change of the position function of the ball at time t in s is less than 0. In other words, the ball is falling at time t. Hence, the precondition axiom for the $bounce(ball, t)$ action is

$$\begin{aligned} &Poss(bounce(ball, t), s) \equiv \\ &\quad t \geq start(s) \wedge position(ball, s)(t) = 0 \wedge \\ &\quad \frac{d}{dt}(position(ball, s))(t) < 0 \end{aligned} \tag{1}$$

where $Poss(a, s)$ means that action a is possible in situation s.

Natural actions, such as $bounce(ball, t)$, are a special case: when a natural action can possibly occur at time t, it *does* occur, unless some other action (perhaps $catch(ball, t)$) occurs sooner. In worlds where no agent has "free will", all actions are natural. Intuitively, an agent has "free will" if it is impossible to predict what actions the agent will perform, or, at least, *when* the agent will perform some action. Nature is taken to be characterized by all of the scientific laws that scientists strive to know, and to the extent that those laws exist and do not change, Nature has no "free will".

In worlds where all actions are natural, if certain conditions are met, it is possible to simulate the evolution of the world. Simulation is possible when in every situation it is possible to determine whether and when actions will occur. In effect, the world evolves deterministically according to the laws of Nature (e.g. Newton's laws). When this is the case, there is only one legal path through the tree of situations.

2.4 The Results of Courses of Action

The ways in which the values of fluents are affected by action occurrences are determined by *successor state axioms*. A formalization of a world includes one successor state axiom for each fluent. The axiom specifies all individual conditions under which the fluent will change, and how the fluent changes under those conditions.

Consider a world where a ball can bounce, and it can be caught, and no other actions can affect it. The successor state axiom for the $position(ball, s)$ fluent would state that if the concurrent action c is possible in s, then the value of $position(ball, do(c, s))$ depends upon what simple actions are in c. If c contains a *bounce* and not a *catch*, the ball's velocity reverses. If c contains a *catch*, the ball's position becomes constant where it is caught. If c contains neither a *bounce* nor a *catch*, the behavior of the ball's position remains unchanged.

The axiom is

$$\begin{aligned} &Poss(c, s) \supset position(ball, do(c, s)) = f \equiv \\ &\quad [(\exists t')bounce(ball, t') \in c \wedge catch(ball, t') \notin c \wedge \\ &\quad time(c) = t_0 \wedge \\ &\quad (\forall t) f(t) = 0 - \frac{d}{dt}(position(ball, s))(t')(t - t_0) - \\ &\qquad 1/2 g(t - t_0)^2] \\ &\quad \vee \\ &\quad [(\exists t')catch(ball, t') \in c \wedge \\ &\quad (\forall t) f(t) = position(ball, s)(t')] \\ &\quad \vee \\ &\quad [(\forall t)bounce(ball, t) \notin c \wedge catch(ball, t) \notin c \wedge \\ &\quad f = position(ball, s)] \end{aligned} \tag{2}$$

where the surface on which the ball bounces is taken to be at position 0, and $g > 0$ is the acceleration due to gravity. Successor state axioms such as this embody Reiter's [Rei91] solution to the frame problem.

3 The Dagstuhl Steam Boiler

In this section, we attempt merely to introduce the Dagstuhl steam boiler problem in enough detail for the purposes of this paper, rather than reproduce the specification. See [Abr94] for the complete specification.

The Dagstuhl steam boiler system consists of the following units:

- the steam boiler with a water level which is kept preferably within the normal operating range, but certainly must be kept within the safe range,
- a device to measure the quantity of water in the steam boiler (denoted by *water_level*),

- four pumps which are either off or on to provide the steam boiler with water (we use *pump*(i) to denote the ith pump),
- one controller to supervise each pump (four in all), reporting on its water flow (we use *pump_control*(i) to denote the ith pump controller),
- a device to measure the quantity of steam exiting the steam boiler (denoted by *steam_rate*),
- an operator desk, from which a *STOP* message can be sent to the controller,
- a message transmission system used for all communication between the controller and the steam boiler components.

We attempt to keep our notation consistent with the original problem description, [Abr94]. Messages are denoted by functions and constants with upper case names. Mandatory messages must be present in every transmission. There are roughly four classes of messages:

Control messages : The controller sends messages to the physical units to direct their actions, and the physical units send messages to the controller to indicate the actual state of the steam boiler.

The control messages sent by the controller to the physical units are

- $VALVE$: sent in *initialization* mode to request the opening and then the closure of the valve for evacuation of water from the steam-boiler,
- $OPEN\_PUMP(n)$: sent to activate *pump*(n),
- $CLOSE\_PUMP(n)$: sent to deactivate *pump*(n).

The control messages sent by the physical units to the controller are

- $PUMP\_STATE(n, b)$: sent to indicate that the state of *pump*(n) is b, which may be 0 or 1, meaning open or closed, mandatory,
- $PUMP\_CNTL\_STATE(n, b)$: sent to indicate that the flow of water from *pump*(n) is b, which may be 0 or 1, meaning there is flow or there is no flow, mandatory,
- $LEVEL(v)$: sent to indicate that the level of water in the boiler is v, mandatory,
- $STEAM(v)$: sent to indicate that steam is exiting the boiler at rate v, mandatory.

Fault Detection and Repair messages : When the controller infers that a component is defective (a pump claims to be operating, but there is no water flow, for example), the controller sends a fault detection message. When the a defective component has been repaired, a repair message is sent to the controller.

The controller sends the following fault detection messages:

- $PUMP\_FAIL\_DETN(n)$: sent (until acknowledgement is received) to indicate that the controller has detected the failure of *pump*(n).
- $PUMP\_CNTL\_FAIL\_DETN(n)$: sent (until acknowledgement is received) to indicate that the controller has detected the failure of *pump_control*(n),
- $LEVEL\_FAIL$: sent (until acknowledgement is received) to indicate that the controller has detected the failure of the water level measuring unit,
- $STEAM\_FAIL$: sent (until acknowledgement is received) to indicate that the controller has detected the failure of the unit that measures the rate of steam exiting the boiler.

The controller receives the following repair messages:

- $PUMP\_REPD(n)$: sent (until acknowledgement is received) to indicate that *pump*(n) has been repaired,
- $PUMP\_CNTL\_REPD(n)$: sent (until acknowledgement is received) to indicate that *pump_control*(n) has been repaired,
- $LEVEL\_REPD$: sent (until acknowledgement is received) to indicate that the water level measuring unit has been repaired,
- $STEAM\_REPD$: sent (until acknowledgement is received) to indicate that the unit that measures the rate of steam exiting the boiler has been repaired.

Acknowledgement messages : The physical units send the controller a corresponding acknowledgement message for each fault detection message. For example, $PUMP\_FAIL\_ACK(n)$ is sent to the controller to acknowledge the receipt of a $PUMP\_FAIL\_DETN(n)$ message. Similarly, the controller sends the physical units a corresponding acknowledgement message for each repair message. For example, $PUMP\_REPD\_ACK(n)$ is sent to the physical units to acknowledge a $PUMP\_REPD(n)$ message.

Other messages : There are also messages that have somewhat administrative purposes.

The controller sends the following administrative messages to the physical units:

- $PROGRAM\_READY$: sent (until acknowledgement is received) in *initialization* mode to indicate that the controller is ready to assume control of the steam boiler,
- $MODE(m)$: sent to to indicate that the program's current mode of operation is m (see

below for the modes of operation), mandatory,

- *EMERGENCY_STOP*: sent to immediately transfer control of the steam boiler to the operators.

The physical units send the following administrative messages to the controller:

- *STOP*: when the controller receives this message three times in a row, the program goes into *emergency_stop* mode,
- *STEAM_BOILER_WAITING*: sent in *initialization* mode to trigger the start of the program,
- *PHYSICAL_UNITS_READY*: sent in *initialization* mode to acknowledge a *PROGRAM_READY* message.

A solution to the steam boiler problem is an implementation of a controller program that will keep the water level in the boiler within the normal operating range by receiving and sending messages through the message system. If for some reason the water level threatens to go outside of the safe operating range, the controller should immediately transmit an *EMERGENCY_STOP* message, and halt.

The operation of the program is cyclic. Every five seconds, the program receives the incoming messages from the steam boiler components. It then computes the set of messages that should be sent out to the components, and transmits those messages. In each cycle, all messages are assumed to be received or transmitted simultaneously.

Components of the steam boiler can become defective. Defective components determine (for the most part) the operation mode of the controller. The operation modes and their corresponding conditions (mostly) are:

initialization : the steam boiler is not yet operating;

normal : no component is defective;

degraded : any component except the water level monitor is defective;

rescue : the water level monitor is defective, but the steam monitor is not. In *rescue* mode, the controller estimates upper and lower bounds on the water level, based on the dynamics of the boiler. If the controller calculates that either bound will be outside the safe operating range at the next cycle, then the water level is considered to be outside the safe range.

emergency_stop : the water level and steam monitors are both defective, or the water level is threatening to go outside the safe range.

When the program detects that a component is faulty, it sends an appropriate message to the operator desk. After the component is repaired, an appropriate message is sent to the controller to inform it of the repair.

4 The Situation Calculus Language

The instantiation of McCarthy's [MH69] situation calculus language used in this paper to formalize the steam boiler controller is due to Reiter [Rei96], largely influenced by Pinto's [Pin94] work on concurrency and continuous processes. The language has the following ontology:

- a sort *situation*, and a distinguished situation constant symbol S_0.
- a sort *time* ranging over the reals.
- a sort *action* of *simple* actions. All actions are instantaneous, and are denoted by a family of functions that take a parameter in the last argument position denoting the time of the action's occurrence. Variables a, a', etc. are used for simple actions.
- a sort *concurrent* of concurrent actions which are sets of simple actions. Variables c, c', etc. are used for concurrent actions.
- a function symbol $time : action \rightarrow \Re$, where $time(a)$ denotes the time of the action a.
- a function symbol $start : situation \rightarrow \Re$, where $start(s)$ denotes the start time of the situation s.
- a function symbol $do : action \times situation \rightarrow situation$.
- The predicate symbol $Poss$, where $Poss(a, s)$ means that the simple action a is possible in situation s (and similarly for any concurrent action c).
- The predicate symbol $<$, where $s < s'$ means that s' is reachable from s through the execution of a sequence of possible actions (simple or concurrent).
- The foundational axioms for the concurrent temporal situation calculus, provided in [Rei96], which are generalizations of those provided in [LR94] and [Rei93] for the nonconcurrent situation calculus. These axioms include unique names axioms for situations, a definition for $<$, a coherency criterion for concurrent actions, and an induction axiom.

4.1 Axiomatizing Application Domains

Levesque *et al.* [LRL+96] list the general types of axioms required to formalize an application domain in the situation calculus. In particular, our axiomatization consists of the following axioms:

- For each *simple* action A, a single action precondition axiom of the form

  $$Poss(A(\vec{x},t),s) \equiv start(s) \leq t \wedge \Phi(\vec{x},t,s)$$

  where $\Phi(\vec{x},t,s)$ is any first order formula with free variables among $\vec{x}$, t, and s whose only term of sort *situation* is s.
- For each fluent f (except for *defined fluents*, which are defined in terms of other fluents), a single successor state axiom. The form of a successor state axiom for a relational fluent is

  $$Poss(c,s) \supset f(\vec{x},do(c,s)) \equiv$$
  $$\gamma_f^+(\vec{x},c,s) \vee f(\vec{x},s) \wedge \neg\gamma_f^-(\vec{x},c,s)$$

  where $\gamma_f^+(\vec{x},c,s)$ and $\gamma_f^-(\vec{x},c,s)$ denote the conditions under which c, if performed in s, results in $f(\vec{x},do(c,s))$ becoming true and false, respectively.

  For a functional fluent, the form of the successor state axiom is

  $$Poss(c,s) \supset f(\vec{x},do(c,s)) = y \equiv$$
  $$\gamma_f(\vec{x},y,c,s) \vee y = f(\vec{x},s) \wedge \neg(\exists y')\gamma_f(\vec{x},y',c,s)$$

  Here, $\gamma_f(\vec{x},y,c,s)$ is a first order formula whose free variables are among $\vec{x},y,c,s$. In the case of Successor State Axiom 2, $\gamma_{position}(\vec{x},f,c,s)$ is

  $$[(\exists t')bounce(ball,t') \in c \wedge catch(ball,t') \notin c \wedge$$
  $$time(c) = t_0 \wedge$$
  $$(\forall t)f(t) = 0 - \tfrac{d}{dt}(position(ball,s))(t')(t-t_0) -$$
  $$1/2g(t-t_0)^2]$$
  $$\vee$$
  $$[(\exists t')catch(ball,t') \in c \wedge$$
  $$(\forall t)f(t) = position(ball,s)(t')]$$
- Unique names axioms for the primitive actions.
- Axioms describing the initial situation.
- The foundational axioms mentioned in the previous section.

5 Formalizing the Steam Boiler Specification

The first step in implementing a steam boiler controller with the situation calculus is to reconcile the inherently procedural operation of a steam boiler controller with the inherently declarative nature of the situation calculus.

The issue is the following. The axiomatizer cannot completely specify what the behavior of the controller will be after it is put into service, because the information comprising the incoming messages that will be received at run-time is not available to her at the time of specification. She cannot write down, for example, that at time t, the controller will receive message m, and do x.

Knowledge-producing actions are required to handle this issue properly (see, for example, [LLL+96, SL93]). The receipt of a message would be represented by the execution of a knowledge-producing action. The axiomatizer would be able to write down that at time t, the controller will receive a message, and if the message is m_1, the controller will do x_1, but if the message is m_2, the controller will do x_2, etc.

In order to keep the particular implementation presented in this paper as simple as possible, we do not use knowledge producing actions. We deal with the issue of incoming messages non-logically, and in order to not completely betray one of the main benefits of the situation calculus approach to the steam boiler problem, namely, that the the approach is logical, we must be careful about these non-logical properties.

We define a predicate $input(m,t)$, which asserts that message m is ready to be received by the controller at time t. We handle incoming messages with a PROLOG `assert` statement, which updates the definition of the $input(m,t)$ predicate as messages are received. When executing a specification that is continually updated in this way, using the `assert` statement, the logical consequences actually used by the controller are all and only those logical consequences that would be used by the controller if all knowledge about future incoming messages were known and specified before the controller is put into service.

The general operation of the situation calculus implementation is as follows:

1. Every five seconds, a set of messages are received from the steam boiler. For each of these messages, m, $Poss(receive(m,t),s)$ is now true at the current time t in the current situation s.
2. The concurrent action consisting of the complete set of $receive(m,t)$ actions is performed, and the values of the fluents are affected accordingly.
3. The new values of the fluents make more actions possible immediately. Since all the actions are natural, any action that is possible (and not prevented by an earlier action) is carried out.
4. The first actions to become possible are those that change the mode of operation of the controller, if any.
5. After the mode has stabilized, a complex action consisting of a set of $transmit(m,t)$ actions will be possible, and carried out.
6. The situation resulting from transmitting messages is stable, and the next actions to be possible are the actions to receive the messages in the next cycle.

In the following sections, we show what such a specification looks like. Rather than present the entire specification, we present enough of the specification to show

the reader how such a specification can be constructed.

5.1 Actions

The following actions are sufficient to model the steam boiler:

- *receive*(m, t): receive message m at time t
- *transmit*(m, t): transmit message m at time t
- *switch_to_mode*(m, t): switch to operation mode m at time t.

These are all natural actions.

5.2 Fluents

The fluents described in this section comprise a sufficient notion of the state of the system. The names of fluents are chosen to be consistent with the original steam boiler specification. The various quantities (e.g. the water throughput of a pump) are considered to be within an upper and lower bound, and each bound has three values associated with it: a *claimed* value, which is the corresponding sensor reading, a *calculated* value, which is calculated by the controller using its knowledge of the dynamics of the boiler, and a *best estimate*, which is the same as the claimed value if the sensor is not defective, and the same as the calculated value if the sensor is defective. Note that the upper and lower bounds on a quantity are both equal to the sensor reading of that quantity if the corresponding sensor is not defective.

mode$(s) = z$: the controller's mode of operation is z, one of {*initialization*, *normal*, *degraded*, *rescue*, *emergency_stop*}

cur_receive_time$(s) = z$: the controller will next receive messages a t time z

waiting_begin_flow$(n, s) = z$: z is the number of transmissions received since pump n was turned on, and pump controller n has not yet confirmed the flow starting (if the pump has not been turned on, $z = -1$)

waiting_stop_flow$(n, s) = z$: z is the number of transmissions received since pump n was turned off, and pump controller n has not yet confirmed the flow ceasing (if the pump has not been turned off, $z = -1$)

$q(s) = z$: the last transmission contained a message claiming the quantity of water in the boiler was z

$v(s) = z$: the last transmission contained a message claiming the quantity of steam exiting the boiler was z

$qc1(s) = z$: z, a function of time, is the calculated lower bound on the quantity of water in the boiler

$qc2(s) = z$: z, a function of time, is the calculated upper bound on the quantity of water in the boiler

$pa1(s) = z$: z is the best estimate of the lower bound on the (constant) throughput of the pumps.

$pa2(s) = z$: z is the best estimate of the upper bound on the (constant) throughput of the pumps.

$pc1(s) = z$: z is the calculated lower bound on the (constant) throughput of the pumps.

$pc2(s) = z$: z is the calculated upper bound on the (constant) throughput of the pumps.

$vc1(s) = z$: z, a function of time, is the calculated minimum rate of steam leaving the boiler

$vc2(s) = z$: z, a function of time, is the calculated upper bound on the rate of steam leaving the boiler

$qa1(s) = z$: z is the best estimate of the lower bound on the quantity of water in the boiler at the start of s

$qa2(s) = z$: z is the best estimate of the upper bound on the quantity of water in the boiler at the start of s

$va1(s) = z$: z is the best estimate of the lower bound on the rate of steam leaving the boiler at the start of s

$va2(s) = z$: z is the best estimate of the upper bound on the rate of steam leaving the boiler at the start of s

transmitted(m, s) : message m was transmitted at the start of s

received(m, t, s) : message m was received at time t in s

transmission_error(s) : a transmission error occurred at the start of s (e.g. a mandatory message was missing from a transmission)

defective(x, s) : component x is defective

steam_boiler_waiting(s) : the steam boiler is waiting for the controller to indicate it is ready to come on-line

program_ready(s) : the program is ready to come on line

physical_units_ready(s) : the steam boiler is ready for control to begin

pump_state(n, b, s) : *pump*(n) has been shut off ($b = 0$), or turned on ($b = 1$)

pump_control_state(n, b, s) : the last transmission contained a message claiming that water from *pump*(n) was ($b = 1$) or was not ($b = 0$) flowing

valve_open(s) : the valve to let water drain out of the steam boiler is open

waiting_ack(m, s) : the controller is waiting for acknowledgement message m

send_ack(x, s) : the controller should send acknowledgement message m in the next transmission

5.3 Initial Situation

The situation is S_0 when the controller is turned on: $mode(S_0) = initialization$, $q(S_0) = 0$, $pa1(S_0)(t) = 0$, $pa2(S_0)(t) = 0$, $v(S_0)(t) = 0$, $\neg defective(x, S_0)$, etc.

5.4 Precondition Axioms

In practice, it is possible to receive a message when that message arrives, and we know that messages will arrive approximately every five seconds. The PROLOG simulator that executes the situation calculus specification is responsible for controlling the timing of the transmit/receive cycle.

To ensure the program detects the absence of a transmission, we invent a message, *tick*, which is guaranteed to be received every five seconds. The value of the $transmission\_error(s)$ fluent is determined by considering what messages were received with each tick. The theorem prover (we use PROLOG) updates the definition of the $input(m,t)$ predicate (using the PROLOG **assert** statement) as messages arrive. The $input(m,t)$ predicate asserts that message m is ready to be received at time t.

The following axiom characterizes when messages are actually received (rather than only ready to be received):

$$\begin{aligned} &Poss(receive(m,t),s) \equiv \\ &\quad t \geq start(s) \wedge t = cur\_receive\_time(s) \wedge \\ &\quad [input(m,t) \vee m = tick] \end{aligned} \tag{3}$$

The precondition axiom for the $transmit(m,t)$ action is such that messages are transmitted exactly when they are appropriate for proper operation of the controller. The axiom ensures that the controller's mode has stabilized before any messages are transmitted. If the controller is in *emergency_stop* mode, only the message $MODE(emergency\_stop)$ should be transmitted. Also, is not possible to transmit a message that was just transmitted. These conditions are dictated by the following defined fluent:

$$\begin{aligned} &transmit\_cond(m,t,s) \equiv \\ &\quad t \geq start(s) \wedge mode(s) \neq emergency\_stop \wedge \\ &\quad \neg[(\exists t')\neg Poss(switch\_to\_mode(m',t'),s)] \wedge \\ &\quad \neg transmitted(m,s) \end{aligned}$$

In addition, the decision to switch the pumps on or off is characterized by the following defined fluent:

$$\begin{aligned} &need\_pumps(s) \equiv \\ &\quad mode(s) \neq initialization \wedge \\ &\quad [[qa1(s) < N_1 \wedge qa2(s) < N_1] \vee \\ &\quad [qa1(s) < N_1 \wedge N_1 < qa2(s) \wedge qa2(s) < N_2]], \end{aligned}$$

where N_1 and N_2 are the upper and lower limits, respectively, of the normal operating range for the water level in the boiler. Control strategies other than this one are possible; however, this one is proposed in [Abr94] as a reasonable candidate.

Given these two defined fluents, the precondition axiom for the $transmit(m,t)$ action is:

$$\begin{aligned} &Poss(transmit(m,t),s) \equiv \\ &\quad m = MODE(emergency\_stop) \wedge t \geq start(s) \wedge \\ &\quad mode(s) = emergency\_stop \wedge \neg transmitted(m,s) \\ &\quad \vee \\ &\quad m = MODE(mode(s)) \wedge transmit\_cond(m,t,s) \\ &\quad \vee \\ &\quad m = PROGRAM\_READY \wedge \\ &\quad transmit\_cond(m,s,t) \wedge \\ &\quad mode(s) = initialization \wedge program\_ready(s) \\ &\quad \vee \\ &\quad m = VALVE \wedge transmit\_cond(m,s,t) \wedge \\ &\quad mode(s) = initialization \wedge \\ &\quad steam\_boiler\_waiting(s) \wedge \\ &\quad [\neg valve\_open(s) \wedge q(s) > N_2 \vee \\ &\quad valve\_open(s) \wedge q(s) > N_1] \\ &\quad \vee \\ &\quad (\exists n)m = OPEN\_PUMP(n) \wedge \\ &\quad transmit\_cond(m,s,t) \wedge \\ &\quad need\_pumps(s) \wedge \\ &\quad [n = 1 \vee n = 2 \vee n = 3 \vee n = 4] \wedge \\ &\quad \neg defective(pump(n),s) \wedge pump\_state(n,0,s) \\ &\quad \vee \\ &\quad (\exists n)m = PUMP\_FAIL\_DETN(n) \wedge \\ &\quad waiting\_ack(PUMP\_FAIL\_ACK(n),s) \wedge \\ &\quad transmit\_cond(m,s,t) \wedge defective(pump(n),s) \\ &\quad \vee ... \\ &\quad \vee \\ &\quad m = STEAM\_REPD\_ACK \wedge \\ &\quad transmit\_cond(m,s,t) \wedge \\ &\quad send\_ack(STEAM\_REPD\_ACK) \end{aligned} \tag{4}$$

The precondition axiom for the $switch\_to\_mode(m,t)$ fluent governs the mode state changes of the controller. The axiom says, for example, that the controller switches to mode *normal* if nothing is defective, and either the mode is *initialization* and the physical units are ready, or the mode is *rescue*, or the mode is *degraded*.

$$\begin{aligned} &Poss(switch\_to\_mode(m,t),s) \equiv \\ &\quad t \geq start(s) \wedge \\ &\quad [m = normal \wedge \neg defective(x,s) \wedge \\ &\quad [mode(s) = initialization \wedge \\ &\quad physical\_units\_ready(s) \vee \\ &\quad mode(s) = rescue \vee mode(s) = degraded] \end{aligned}$$

$$
\begin{aligned}
&\vee \\
&m = degraded \wedge \\
&[mode(s) = normal \vee mode(s) = rescue] \wedge \\
&[(\exists x) defective(x, s)] \wedge \\
&[(\forall x) defective(x, s) \supset x \neq water\_level] \\
&\vee \\
&m = rescue \wedge \\
&[mode(s) = degraded \vee mode(s) = normal] \\
&\wedge defective(water\_level, s) \\
&\vee \\
&m = emergency\_stop \wedge \\
&[mode = initialization \wedge \\
&[defective(steam\_rate, s) \vee \\
&defective(water\_level, s)] \vee \\
&transmission\_error(s) \vee water\_level\_risk(s) \vee \\
&mode(s) = rescue \wedge [defective(steam\_rate, s) \vee \\
&(\exists n) defective(pump\_control(n), s)]]] \qquad (5)
\end{aligned}
$$

5.5 Successor State Axioms

We present just the most interesting, representative successor state axioms.

The successor state axiom for the $mode(s) = m$ fluent is straightforward. The only action that can affect the mode of the program is a $switch\_to\_mode(m, t)$ action:

$$
\begin{aligned}
&Poss(c, s) \supset mode(do(c, s)) = m \equiv \\
&\quad switch\_to\_mode(m, time(c)) \in c \vee \qquad (6) \\
&\quad switch\_to\_mode(m', time(c)) \notin c \wedge mode(s) = m
\end{aligned}
$$

The successor state axiom for $defective(x, s)$ characterizes the conditions under which a component is considered defective. It says, among other things, that $pump(n)$ is defective if it changes state spontaneously, or if an $OPEN\_PUMP(n)$ or $CLOSE\_PUMP(n)$ message were sent in the previous cycle, but the flow has not yet started or stopped, respectively.

$$
\begin{aligned}
&Poss(c, s) \supset defective(x, do(c, s)) \equiv \\
&\quad (\exists n) x = pump(n) \wedge \\
&\quad [[[(\exists b) receive(PUMP\_STATE(n, b), time(c)) \in c \\
&\quad \wedge (\exists a) pump\_state(n, a, s) \wedge a \neq b] \vee \\
&\quad pump\_state(n, 0, s) \wedge \\
&\quad waiting\_begin\_flow(n, s) = 1 \vee \\
&\quad pump\_state(n, 1, s) \wedge \\
&\quad waiting\_stop\_flow(n, s) = 1] \vee \\
&\quad receive(PUMP\_STATE(n, b), time(c)) \notin c \wedge \\
&\quad defective(x, s)] \\
&\quad \vee \ldots \qquad (7)
\end{aligned}
$$

5.6 Defined Fluents

Defined fluents are fluents that are defined in terms of other fluents. We could write down a successor state axiom for any defined fluent. It would be a compilation of the successor state axioms of the fluents in the definition of the defined fluent that have a situation argument of the form $do(c, s)$. However, the resulting successor state axiom would be larger and more complicated than the defined fluent.

The defined fluent $qa2(s) = z$ means that z is the best estimate of the upper bound on the quantity of water in the boiler. If the water level detection equipment is not defective, the program accepts the value given by that equipment as the best estimate; otherwise, it uses the value calculated by $qc2(s)(t)$ at the time of the action which started the situation. The formula for $qa1$ is similar to this one for $qa2$:

$$
\begin{aligned}
&qa2(do(c, s)) = z \equiv \\
&\quad defective(water\_level, s) \wedge z = qc2(s)(time(c)) \vee \\
&\quad \neg defective(water\_level, s) \wedge z = q(do(c, s)) \qquad (8)
\end{aligned}
$$

6 The Situation Calculus Simulator

In this section we discuss a PROLOG technology simulator. The simulator can simulate a concurrent situation calculus specification like the steam boiler formalization presented in the previous section.

6.1 The PROLOG Simulator

A situation calculus model defines a tree of situations emanating from the distinguished situation S_0. Some of the situations in the tree correspond to *legal* situations, and some do not. A legal situation is consistent with the laws of Nature, in that a natural action must occur at the time dictated by natural laws governing the behavior of the system, unless the action is prevented from occurring by an earlier action. Reiter [Rei96] defines the $legal(s)$ predicate to formalize this principle:

$$
\begin{aligned}
&legal(s) \equiv \\
&\quad S_0 \leq s \wedge \\
&\quad (\forall a, c, s').natural(a) \wedge Poss(a, s') \wedge \qquad (9) \\
&\quad do(c, s') \leq s \supset a \in c \vee time(c) < time(a).
\end{aligned}
$$

Here, $\leq$ is the ordering relation defined by the foundational axioms mentioned earlier. The *legal* predicate is instrumental in the implementation of a simulator, as will become clear.

A domain of discourse in which all actions are natural is said to comply with Reiter's [Rei96] Natural World Condition (NWC). This condition assures a deterministic simulation.

Another concept crucial to the implementation of a simulator is the notion of Reiter's [Rei96] Least Natural Time Points:

$$lntp(s,t) \equiv \quad (10)$$
$$(\exists a)[natural(a) \wedge Poss(a,s) \wedge time(a) = t] \wedge$$
$$(\forall a)[natural(a) \wedge Poss(a,s) \supset time(a) \geq t].$$

Informally, the least natural time point is the earliest time at which any natural action can possibly occur in a situation. The Least Natural Time Point Condition ($LNTPC$) is the following:

$$(\forall s).(\exists a)[natural(a) \wedge Poss(a,s)] \supset (\exists t) lntp(s,t). \quad (11)$$

This condition states that every situation in which there is a possible natural action has a least natural time point. An example of a world where this condition fails is one in which we have $(\forall a).natural(a) \equiv (\exists x,t)a = B(x,t)$, where x ranges over the non-zero natural numbers, and $Poss(B(x,t),s) \equiv t = start(s) + 1/x$.

Reiter [Rei96] puts this all together with his foundational axioms for the concurrent temporal situation calculus and proves:

$$LNTPC \wedge NWC \supset legal(do(c,s)) \equiv$$
$$\{legal(s) \wedge Poss(c,s) \wedge start(s) < time(c) \wedge$$
$$(\forall a)[a \in c \equiv Poss(a,s) \wedge lntp(s,time(a))]\} \quad (12)$$

Formula 12 is the engine for the simulator. The simulator is a PROLOG procedure that takes a situation term s as an argument (initially S_0), prints its argument, constructs a set of actions c such that

$$(\forall a)[a \in c \equiv Poss(a,s) \wedge lntp(s,time(a))],$$

and recursively calls itself with $do(c,s)$. In so doing, the simulator follows the path of legal situations (there is only one path of legal situations when all actions are natural), simulating the evolution of the system.

Here is the PROLOG code:

```
simulate(S) :-
    nl,nl,print(S),
    setof(A,lntp(S,A),C),
    simulate(do(C,S)).

lntp(S,A):-
    natural(A)),
    poss(A,S),
    time(A,T),
    not (
         natural(A_prime),
         poss(A_prime,S),
         time(A_prime,T_prime),
         T>T_prime
        ).
```

The version of this code for use with the steam boiler controller needs to add information about incoming messages as the simulation proceeds, so the `simulate(S)` procedure becomes:

```
simulate(S) :-
      setof(A,lntp(S,A),C),
      transmit_messages(C),
      simulate(do(C,S))
      ;
      read(Transmission),
      receive_messages(Transmission),
      simulate(S).
```

The `transmit_messages(C)` procedure actually transmits any messages, m, from $transmit(m,t)$ actions that are present in the concurrent action, C. When no more natural actions are possible (when the `setof(A,lntp(S,A),C)` procedure fails), the controller process is blocked by the `read(Transmission)` procedure until a transmission arrives from the steam boiler. Upon receiving a new set of messages, more natural actions are enabled, and control of the steam boiler continues.

6.2 Translating a Model to PROLOG

Clark's completion semantics for logic programming ([Cla78]) admit a translation from the situation calculus axioms to PROLOG clauses. The procedure is to simply make the implication in the axioms go only one way, and write down the clausal form. For example, the PROLOG equivalent of Axiom 6 is

```
mode(do(C,S),M):-
    poss(C,S),
    (member(switch_to_mode(M,T),C),
     ;
     not member(switch_to_mode(M_prime,T),C),
     mode(S,M)).
```

6.3 Simulation

The situation calculus steam boiler controller is capable of controlling the FZI simulation ([Lot95]) of the Dagstuhl steam boiler. The following situation term (read from bottom to top) denotes the course of action associated with initialization and startup. After the level of water drops below 400, the pumps are turned on. At time 20 water is flowing at the pump controllers, and the level of water in the boiler is rising due to the influx of water.

```
do([receive(level(443.0),20),
    receive(steam(12.0),20),
    receive(PUMP_STATE(1,1),20),...,
    receive(PUMP_CONTROL_STATE(1,1),20)],
do([transmit(MODE(NORMAL),15)],
do([receive(level(375.5),15),
    receive(steam(7.0),15),
    receive(PUMP_STATE(1,1),15),...,
    receive(PUMP_CONTROL_STATE(1,0),15)],
do([transmit(MODE(NORMAL),5),
    transmit(OPEN_PUMP(1),10),...,
```

```
    transmit(OPEN_PUMP(4),10)],
do([receive(level(398.0),10),
    receive(steam(2.0),10),
    receive(PUMP_STATE(1,0),10),...,
    receive(PUMP_CONTROL_STATE(1,0),10)],
do([transmit(MODE(NORMAL),5)],
do([switch_to_mode(normal,5)],
do([receive(PHYSICAL_UNITS_READY,5),
    receive(level(400.0),5),
    receive(steam(0.0),5),
    receive(PUMP_STATE(1,0),5),...,
    receive(PUMP_CONTROL_STATE(1,0),5)],
do([transmit(PROGRAM_READY,0),
    transmit(MODE(INITIALIZATION),0),
do([receive(STEAM_BOILER_WAITING,0)],s0)...)
```

7 Evaluation of the Solution

In this section, we attempt to apply the evaluation criteria given in [ABL95] to this formalization and simulation. These criteria were formulated by participants at the Dagstuhl meeting to evaluate various solutions to the steam boiler problem and to compare the specific merits and drawbacks of the formal methods used in those solutions. We paraphrase the criteria in the following subsections.

7.1 Formality and Rigor

Is the requirements specification of the solution formal and rigorous, and can the functional and architectural designs be formally and rigorously verified against the specification?

The situation calculus solution is especially strong in this regard, since the specification (at least implicitly) constitutes a functional and architectural design, and it is an implementation, executable on a provably correct simulator.

Other solutions to the problem are based on logical formalisms, but those formalisms are separate from the actual implementation. For example, Leßke and Merz ([LM95]) use Lamport's Temporal Logic of Actions (TLA) [Lam94] in their solution, leaving the implementation language open. They consider this a feature, since the implementation language can be chosen to suite a variety of machine architectures. Rischel *et al* ([RCM+95]) use a specification language that has a real-time interval logic semantics for the specification, and they use C code for the implementation.

7.2 Practicality

Has the solution actually been implemented and can it control the FZI simulation [Lot95] of the steam boiler?

The situation calculus solution is also strong in this regard, since an elementary change in syntax of the specification produces a working implementation. Many of the other solutions produced working implementations.

7.3 Readability and Ease of Use

What level of specialized knowledge is required to read and work with the formalism? How much effort did the solution to the problem require?

The situation calculus is essentially a standard first-order predicate calculus language. Hence, it is conceivable that anyone familiar with the predicate calculus would be able to read the language, even without any prior situation calculus experience.

On the other hand, the situation calculus steam boiler solution is at the implementation level. Other than the general form of a situation calculus specification, the situation calculus has not yet been associated with any well defined methodology involving higher-level abstraction, stepwise refinement, or modularization. This is in contrast to the many of the other solutions, whose formalisms have been designed, at least in part, with ease of use and readability as a primary concern, often as part of a specification methodology.

One such methodology is GrafTab ([SI95]), which uses graphs to describe the flow of the system states, and tables to specify the state condition and actions in the states. A GrafTab solution consists of four distinct documents.

Although readability and ease of use are not the primary design goals of the situation calculus, modeling methodologies and software programs to aid in the development of situation calculus specifications might well receive the attention of researchers in the future.

Acknowledgements

Ray Reiter provided useful comments and important corrections. I have also benefited from discussions with Javier Pinto.

References

[ABL95] Jean-Raymond Abrial, Egon Boerger, and Hans Langmaack. Preliminary report for the Dagstuhl-seminar 9523: Methods for semantics and specification. Available via WWW at http://www.informatik.uni-kiel.de/~procos/dag9523/dag9523.html, 1995.

[Abr94] Jean-Raymond Abrial. Steam-boiler control specification problem. Distributed to the participants of the Dagstuhl Meeting, "Methods for Semantics and Specification", June 4-9, 1995. This paper,

and its companion, "Additional information concerning the physical behavior of the steam boiler", are available via WWW at http://www.informatik.uni-kiel.de/~procos/dag9523/dag9523.html, August 1994.

[Cla78] K. L. Clark. Negation as failure. In Hervé Gallaire and Jack Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, New York, 1978.

[Kel96] T. G. Kelley. Reasoning about physical systems with the situation calculus. In *COMMON SENSE '96: the third symposium on logical formalizations of commonsense reasoning*, Stanford University, January 1996.

[Kui86] Benjamin Kuipers. Qualitative simulation. *Artificial Intelligence*, 29:289–338, 1986.

[Lot95] Annette Lötzbeyer. Simulator for the dagstuhl seminar 1995: Steam boiler control specification problem. The README file and tcl/tk source code for the simulator are available via WWW at http://www.informatik.uni-kiel.de/~procos/dag9523/dag9523.html, 1995.

[Lam94] Leslie Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, 1994.

[LLL+96] Yves Lespérance, Hector J. Levesque, Fangzhen Lin, Daniel Marcu, Raymond Reiter, and Richard B. Scherl. Foundations of a logical approach to agent programming. In M. Wooldridge, J. P. Müller, and M. Tambe, editors, *Intelligent Agents Volume II—Proceedings of the 1995 Workshop on Agent Theories, Architectures, and Languages (ATAL-95)*, pages 331–346. Springer-Verlag, Lecture Notes in Artificial Intelligence, 1996. To Appear.

[LM95] Frank Leßke and Stephan Merz. Steam boiler control specification problem: A tla solution. This paper is available via WWW at http://www.informatik.uni-kiel.de/~procos/dag9523/dag9523.html, September 1995.

[LR94] F. Lin and R. Reiter. State constraints revisited. *Journal of Logic and Computation*, 4(5):655–678, 1994.

[LRL+96] H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. Scherl. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming, Special Issue on Reasoning about Action and Change*, 1996. To appear.

[MH69] John McCarthy and Patrick 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, Scotland, 1969.

[Pin94] Javier Andrés Pinto. *Temporal Reasoning in the Situation Calculus*. PhD thesis, University of Toronto, Toronto, Ontario, Canada, February 1994.

[RCM+95] Hans Rischel, Jorge Cuellar, Simon Mørk, Anders P. Ravn, and Isolde Wildgruber. Development of safety-critical real-time systems. This paper is available via WWW at http://www.informatik.uni-kiel.de/~procos/dag9523/dag9523.html, 1995.

[Rei91] 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.

[Rei93] R. Reiter. Proving properties of states in the situation calculus. *Artificial Intelligence*, 64:337–351, 1993.

[Rei96] 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)*. Morgan Kaufmann Publishers, San Francisco, CA, 1996.

[San89] Erik Sandewall. Combining logic and differential equations for describing real-world systems. In *First International Conference on Principles of Knowledge Representation and Reasoning*, San Mateo, CA, 1989. Morgan Kaufmann.

[SI95] So-Ming So and Mabo R. Ito. Graftab — an innovative requirements specification method. This paper is available via WWW at http://www.informatik.uni-kiel.de/~procos/dag9523/dag9523.html, October 1995.

[SL93] Richard B. Scherl and Hector J. Levesque. The frame problem and knowledge-producing actions. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 689–695, Washington, DC, July 1993. AAAI Press/The MIT Press.

Actions and Events

The PMA Revisited

Andreas Herzig
IRIT, Université Paul Sabatier
118 route de Narbonne, F-31062 Toulouse Cédex, France
herzig@irit.fr
http://www.irit.fr/ACTIVITES/EQ_ALG/Herzig/home.html
Fax: (+33) 6155.8325

Abstract

The so-called Possible Models Approach (PMA) is a minimal change semantics for database updates. In this paper we give a sound and complete decision procedure for the PMA which is based on normal forming. We prove complexity results in particular for the case of updates restricted to conjunctions or disjunctions of literals. Finally, we sketch how integrity constraints can be taken into account.

1 INTRODUCTION

The Possible Models Approach (PMA for short) of Winslett (1988, 1990, 1995) is one of the most popular and successful semantical approaches in the domain of database updates. From the formal point of view, the PMA has a beautiful mathematical structure, with a notion of minimal change based on a simple closeness criterion between interpretations. Nevertheless, it has been criticized as being inadequate in particular in applications where integrity constraints appear.

The aim of this paper is twofold: first, we show a new way to correctly handle integrity constraints. Second we give a decision procedure for the PMA. Its core is a set of rewrite rules that allows us to transform formulas to formulas of classical logic. In this way theorems can be proved in a two-step procedure: first put the formula in normal form, and then apply a decision procedure for classical logic.

The rest of the paper is organized as follows: in section 2 we review the PMA, and in section 3 we reformulate it in conditional logic. In section 4 we present the decision procedure and give examples. Then we prove termination (section 5) and correctness (section 6) of the algorithm, and give complexity results (section 7). Finally, in section 8 we sketch how integrity constraints can be integrated into our presentation of the PMA.

2 THE PMA

2.1 The Original Semantics

We present the PMA semantics as described in (Winslett 1988, 1990, 1995).

We suppose a standard language of classical logic built from a set of atoms $ATM = \{p, q, r, \ldots\}$ with the classical connectives $\wedge, \vee, \neg, \top, \bot$. Formulas are denoted by $A, B, C, \ldots$. We consider $A \rightarrow B$ and $A \leftrightarrow B$ to be abreviations of $\neg A \vee B$ and $(A \rightarrow B) \wedge (B \rightarrow A)$.

We view interpretations as subsets of the set of atoms ATM. Given a formula A, $|A| = \{w \subseteq ATM : w \models A\}$ is the set of interpretations satisfying A.

Definition 1 *Given two interpretations $w, v \subseteq ATM$, the distance between w and v is the symmetric difference*

$$w \dot{-} v = (w - v) \cup (v - w)$$

In other words, $w \dot{-} v$ is the set of atoms whose interpretation differs: $w \dot{-} v = \{p :$ either $p \in w$ and $p \notin v$, or $p \notin w$ and $p \in v\}$. E.g. for $w = \{p, q, r\}$ and $v = \{p, r, t\}$ we have $w \dot{-} v = \{q, t\}$.

The update of an interpretation w by a set of interpretations U is the set of those elements of U which are closest, i.e. whose distance to w is minimal.

Definition 2 *Let $w \subseteq ATM$ and $U \subseteq 2^{ATM}$.*
$w \star U =$

$$\{u \in U : \textit{there is no } u' \in U \textit{ such that } w \dot{-} u' \subset w \dot{-} u\}$$

E.g. $w \star \emptyset = \emptyset$, i.e. updating with the empty set (which is the interpretation of a contradiction) results in the empty set, and $\emptyset \star \{\{p\}, \{q\}, \{p, q\}\} = \{\{p\}, \{q\}\}$.

Slightly abusing notation we define the update of a set of interpretations.

Definition 3 *Let* $U, V \subseteq 2^{ATM}$.

$$U \star V = \bigcup_{w \in U} w \star V$$

In the PMA, the update $KB \star A$ of a database KB by a formula A is identified with the set of interpretations $|KB| \star |A|$.

Now under the standard notion of semantical consequence we have

$$KB \star A \models C \text{ iff } |KB| \star |A| \subseteq |C|$$

2.2 Two Criticisms

The first criticism concerns the interpretation of disjunction in the PMA, and the second one the implementation of integrity constraints.

Suppose we are given the database $KB = \{\neg p, \neg q\}$. Its update by $p \vee q$ is

$$|\{\neg p, \neg q\}| \star |p \vee q| = \{\emptyset\} \star \{\{p\}, \{q\}, \{p, q\}\}$$

$$= \{\{p\}, \{q\}\} = |\{p \vee q, \neg p \vee \neg q\}|$$

Thus the result of updating KB by the inclusive or is the exclusive or. There are several counterexamples where such an exclusive interpretation of the inclusive or is inadequate, and the PMA has been criticized for that (Eiter et al. 1992).

Second, suppose we are given a finite set of integrity constraints IC (that is confused with the conjunction of its elements). We must define an update operation $\star_{IC}$ which takes into account such integrity constraints. It has often been proposed (see e.g. (Katsuno and Mendelzon 1991)) to define such an operation by $KB \star_{IC} A = KB \star (A \wedge IC)$. This amounts to selecting the closest $A \wedge IC$-interpretations.

By means of the following counterexample it has been shown by Ginsberg (see e.g. (Lifschitz 1990)) that such a proposal is problematic in particular if $\star$ is the PMA update operator: let up_1 mean 'switch 1 is up', up_2 'switch 2 is up', and l 'the light is on'. Suppose there is a circuit such that the light is on exactly when both switches are in the same position. Hence

$$IC = (up_1 \leftrightarrow up_2) \leftrightarrow l$$

Let $KB = \{up_1, up_2, l\}$. Then one would expect that $KB \star (\neg up_1 \wedge IC) \models \neg l$. As well, we would expect $KB \star (\neg up_1 \wedge IC) \models up_2$, i.e. the second switch does not move. It turns out that neither is the case in the PMA.

We respond to these criticisms in section 8. In particular we shall propose a new, more sophisticated way of implementing integrity constraints.

2.3 The Existing Decision Procedures

Several decision procedures have been proposed for the PMA. The first one has been given in (Chou and Winslett 1991, 1994, Winslett 1995). Others can be found in (Del Val 1992) and (Grahne and Mendelzon 1995).

In the approaches of Chou and Winslett and of Grahne and Mendelzon, the algorithm directly manipulates models of the database KB. This amounts to enumerating the models of the database and updating each model.

As noted in (Del Val 1992), such procedures might be adequate in situations of almost complete information, in which there are only a few models. They clearly run into difficulties in cases where information is usually incomplete, as it is the case in many applications such as robotics and more generally commonsense reasoning. In such contexts, the number of models can be exponential w.r.t. the size of the database.

The approach of Del Val manipulates partial models of the database KB. His procedure enumerates the minimal models and updates each of them. Although he claims it to be a syntactical characterization of the PMA, it has a strong model-based flavour: in the basic mechanism, with the database and the update formula in disjunctive normal form, his algorithm uses a function *Diff(.,.)* computing the difference between two conjunctions of literals, which is basically the symmetric difference function $\dot{-}$ of the semantics.

All these methods are in terms of an update operator (involving a database and an update) and not in terms of a conditional operator (involving an update and a goal). This leads to procedures operating on models of the database, and makes their complexity exponential in the size of the database. Our procedure for the PMA will avoid at least that type of exponential complexity. The price to pay for that is exponential increase in the size of the update formula. This comes with putting the update in disjunctive normal form (which can be viewed as working with partial models of the update). Nevertheless, as databases tend to be big and updates to be small, our procedure should be preferrable in practical applications. As well, we shall give complexity results that are not in the above

papers for the cases of updates by conjunctions or disjunctions of literals.

3 A REFORMULATION IN CONDITIONAL LOGIC

In this section we give an equivalent presentation of the PMA in terms of conditionals. This has been done before in (Fariñas and Herzig 1988) for the case where updates are restricted to literals, and in (Fariñas and Herzig 1994c, 1997) for the general case. In both papers, the PMA has been characterized axiomatically. In the more general setting of update operations satisfying the postulates of Katsuno and Mendelzon (1991) this has been done in (Grahne 1991, Eiter and Gottlob 1992, Ryan and Schobbens 1996, 1997).

The reason for such a presentation of the PMA is that our decision procedure introduces formulas with nested conditional operators (which cannot be expressed equivalently with update operators). As well, it becomes clear that the PMA is a particular case of Burgess's (1981) semantics for conditional logics.

The semantical apparatus is the same as before, based on the update $w \star U$ of an interpretation w by a set of interpretations U. But at the syntactical level, instead of re-using the update operator $\star$ we introduce a conditional operator $>$. The conditional $A > C$ is read as a hypothetical update: "if the current database is updated with A then C follows".

We have a standard language of conditional logic resulting from the addition of a conditional operator $>$ to the language of classical logic. In the formula $A > C$, A is called the *hypothesis* and C the *consequence*. Note that nested conditional operators as in $(p > q) > r$ are allowed. We call a formula *classical* if no conditional operator occurs in it. We stipulate that $\neg$ binds stronger than $>$, which in turn binds stronger than the rest of the connectives and associates to the right. E.g. $p > \neg q > r \vee t$ is $(p > ((\neg q) > r)) \vee t$.

We need a function *atm* which associates to every formula the set of atoms appearing in it.

Definition 4 *Let A be a formula.*

$$atm(A) = \{p \in ATM : p \text{ occurs in } A\}$$

E.g. $atm(\bot) = \emptyset$, and $atm(p \rightarrow (r \wedge q)) = \{p, q, r\}$. Now we inductively define the extension $|.|$ of a formula, which is a set of interpretations.

Definition 5 *Let A and C be formulas.*

$$|A > C| = \{w \subseteq ATM : w \star |A| \subseteq |C|\}$$

For the other connectives, $|.|$ is defined as in classical logic.

As usual, semantical consequence $KB \models C$ is defined by $|KB| \subseteq |C|$.

Remark In this way, the PMA is a particular case of Burgess's (1981) relational semantics, in the sense that its semantics corresponds to a unique "big" model. In that model, every classical interpretation is identified with some world. To see that, it is sufficient to prove that every relation $R(w) = \{(w_1, w_2) : w_1 \dot{-} w \subseteq w_2 \dot{-} w\}$ is a partial preorder, i.e. a reflexive and transitive relation. Note that the PMA-semantics is not an instance of Lewis's sphere models, because the preorder $R(w)$ is only partial and not total.

Finally we formally link the two presentations of the PMA by the so-called Ramsey Test (Gärdenfors 1988, Grahne 1991):

Proposition 1 *Let KB, A and C be classical. Then $KB \star A \models C$ iff $KB \models A > C$.*

Proof $(KB \star A) \models C$ means $|KB| \star |A| \subseteq |C|$. This is equivalent to $w \star |A| \subseteq |C|$ for every $w \in |KB|$. But this is nothing else than $w \in |A > C|$ for every $w \in |KB|$, which means $KB \models A > C$ in our conditional logic presentation.

4 THE DECISION PROCEDURE

In this section we show that every formula can be rewritten to a classical formula. This gives us the decision procedure.

First of all we recall some notions: as usual, a *literal* is an atom or the negation of an atom. Literals are denoted by L, L', L_1, ... Note that $\top$ and $\bot$ are not literals. *Clauses* are disjunctions of literals, and *cubes* are conjunctions of literals. E.g. $p \vee q \vee \neg p$ is a clause, and $p \wedge \neg r \wedge t$ is a cube.

A classical formula is in *negation normal form* (NNF for short) if it is constructed from literals, $\top$ and $\bot$ with conjunction and disjunction. In other words, for a formula in NNF, negations only govern atoms. A classical formula is in *disjunctive normal form* (DNF for short) if it is either $\top$ or $\bot$, or a non-tautological disjunction of consistent cubes (i.e. every cube does not contain both an atom and its negation). E.g. $p \wedge q$ and $p \vee q$ are in DNF, and neither $p \vee \neg p$ nor $p \wedge \bot$ nor $p \vee \bot$ nor $(p \wedge q \wedge \neg p) \vee r$ are in DNF.

Lemma 1 *For every classical formula C there is a classical formula C' such that $C \leftrightarrow C'$, $atm(C) = atm(C')$, and C' is in DNF.*

A key notion will be that of interference:

Definition 6 *A and B interfere iff $atm(A) \cap atm(B) \neq \emptyset$.*

Hence two formulas interfere if they have some atom in common. E.g. $p \vee q$ and $r \wedge (\neg q \vee t)$ interfere, and $p \vee q$ and $r \wedge t$ don't.

Next we state a useful property of classical logic saying that for every formula of classical logic, we can always 'factor out' some A which does not interfere with the rest of the formula.

Lemma 2 *Let C be a classical formula with $atm(C) \neq \emptyset$. Then there is a classical formula C' such that $C \leftrightarrow C'$, $atm(C) = atm(C')$, and*

- *C' is of the form $(A \wedge B) \vee (\neg A \wedge B') \vee B''$,*
- *A and B, A and B', and A and B'' do not interfere.*

Proof One can apply the following algorithm: First put the hypothesis in DNF. Whenever there is a cube containing both p and $\neg p$ as conjuncts, replace that cube by $\bot$. Choose some atom p occurring in the DNF. (There is at least one because $atm(C) \neq \emptyset$.) As by the definition of the DNF there is no cube with p occurring both positively and negatively in it, factoring p and $\neg p$ we can put the formula into the form

$$(p \wedge B) \vee (\neg p \wedge B') \vee B''$$

such that p does not occur in B, B' and B'' any longer (note that B, B', and B'' might be $\top$ or $\bot$). Hence p and B, p and B', and p and B'' do not interfere.

4.1 The Normal Forming Algorithm

Here comes the normal forming algorithm which rewrites every conditional with classical hypothesis and consequence into a classical formula. This algorithm will be iterated in the decision procedure.

$A, B, \ldots$ denote general formulas, $L, L_1, \ldots$ literals, and $p, q, \ldots$ atoms.

procedure eliminate-cond
input: a conditional $A > C$ such that A and C are classical
output: a formula G that is classical

1. put A in DNF and C in NNF

2. apply the following rewrite rules:

a. **if** $n \geq 2$ **then**

$$(L_1 \wedge \ldots \wedge L_n) > C' \rightsquigarrow L_1 > (\ldots > (L_n > C') \ldots)$$

b. **if** $n \geq 2$ **then**
$(L_1 \vee \ldots \vee L_n) > C' \rightsquigarrow$

$$((L_1 \vee \ldots \vee L_n) \wedge C') \vee (L_1 > C' \wedge \ldots \wedge L_n > C')$$

c. **if** $\text{card}(atm(A')) \geq 2$, A' is neither a clause nor a cube, and A' can be transformed by factoring [1] into $(p \wedge B) \vee (\neg p \wedge B') \vee B''$ with p and B, p and B', and p and B'' not interfering and $atm(A') = atm((p \wedge B) \vee (\neg p \wedge B') \vee B'')$, **then**

$A' > C' \rightsquigarrow$

$$(p \wedge (B \vee B'') > C' \wedge (B \vee B' \vee B'') > (B \vee B'' \vee \neg p > C')) \vee$$

$$(\neg p \wedge (B' \vee B'') > C' \wedge (B \vee B' \vee B'') > (B' \vee B'' \vee p > C'))$$

d. **if** $\text{card}(atm(A')) \geq 2$, A' is neither a clause nor a cube, and A' can be transformed by factoring into $(L \wedge B) \vee B'$ with L and B, and L and B' not interfering and $atm(A') = atm((L \wedge B) \vee B')$ **then**

$A' > C' \rightsquigarrow$

$$(L \wedge (B \vee B') > C') \vee (\neg L \wedge B' > C' \wedge (B \vee B') > (B' \vee L > C'))$$

3. apply the following rewrite rules:

a. $\top > C' \rightsquigarrow C'$

b. $\bot > C' \rightsquigarrow \top$

c. $L > (C_1 \wedge C_2) \rightsquigarrow L > C_1 \wedge L > C_2$

d. $L > (C_1 \vee C_2) \rightsquigarrow L > C_1 \vee L > C_2$

e. $L > L \rightsquigarrow \top$

f. $p > \neg p \rightsquigarrow \bot$

g. $\neg p > p \rightsquigarrow \bot$

h. **if** $atm(L) \neq atm(L')$ **then** $L > L' \rightsquigarrow L'$

i. $L > \top \rightsquigarrow \top$

j. $L > \bot \rightsquigarrow \bot$

k. $A' \wedge \top \rightsquigarrow A'$

l. $\top \wedge A' \rightsquigarrow A'$

m. $A' \wedge \bot \rightsquigarrow \bot$

n. $\bot \wedge A' \rightsquigarrow A'$

[1] v. Lemma 2

Remarks In step 1, if $atm(A') = \emptyset$ then A' is replaced by either $\top$ or $\bot$, and if $atm(A') = \{p\}$ then A' is replaced by either p or $\neg p$. In step 2.a, the hypothesis is a single cube, and as we are in DNF, the cube is consistent. In step 2.b, the hypothesis is a clause, and as we are in DNF, the clause is non-tautological.

Iterating step 2 we eliminate all conditionals $A' > C'$ with classical hypothesis such that that A' is "complex", i.e. such that hypotheses are of the form $\top$, $\bot$, $L_1 \wedge \ldots \wedge L_n$, or $L_1 \vee \ldots \vee L_n$. This means that at the end of step 2 we have $atm(A') \leq 1$. Note that in step 2.c, as $atm(A') \geq 2$, factoring out some p is always possible by Lemma 2. Moreover, as $\text{card}(atm(A')) \geq 2$, we must have $\text{card}(atm(B) \cup atm(B') \cup atm(B'')) \geq 1$. (Note also that nested conditionals are introduced here.) Step 2.d is a simplification of step 2.c for the case where either B or B' is $\bot$.

Iterating step 3.a and 3.b we eliminate all conditionals $A' > C'$ with a classical hypothesis A' such that $\text{card}(atm(A')) \leq 1$. Iterating steps 3.c - d we distribute the literal hypotheses until every conditional with a classical hypothesis is of the form $L > L'$, where L and L' are literals. Steps 3.k - n correspond to standard simplification rules of classical logic.

Iterating step 3.a - n we eliminate all conditionals $A' > C'$ with a classical hypothesis A' such that $\text{card}(atm(A')) \leq 2$.

4.2 The Decision Procedure

Now it is easy to build a procedure from the normal forming algorithm which decides whether a given formula is valid in the PMA. The story goes as follows:

procedure decide
input: a formula F
output: { *valid, non-valid* }

1. **while** there is a conditional operator occurring in F **do**
   (a) choose some conditional $A > C$ with A and C classical
   (b) apply the normal forming algorithm to $A > C$ resulting in some G
   (c) in the formula F, replace $A > C$ by G
2. apply some decision procedure for classical logic to F
3. **if** F is classically valid **then** return *valid*, **else** return *non-valid*

Remark In practice we want to know whether a formula C follows from the update of a classical database KB by a classical formula A. In this case there is a slightly better procedure: first, rewrite the conditional $A > C$ into an equivalent classical formula G by the normal forming algorithm. Then use a decision procedure for classical logic to check whether $KB \models G$.

4.3 Examples

Example Let $A = p \rightarrow (q \wedge \neg p) > (q \wedge \neg p)$. First, the conditional $(q \wedge \neg p) > (q \wedge \neg p)$ is chosen and is rewritten in step 2.a of **eliminate-cond** to $q > \neg p > (q \wedge \neg p)$.

Then from $p \rightarrow q > \neg p > (q \wedge \neg p)$, the conditional $\neg p > (q \wedge \neg p)$ is chosen and rewritten to $\neg p > q \wedge \neg p > \neg p$ in step 3.c. In step 3.e, this is rewritten to $\neg p > q \wedge \top$ and then in step 3.h to $q \wedge \top$.

Now the resulting $p \rightarrow q > (q \wedge \top)$ is first put into DNF in step 1 and becomes $p \rightarrow q > q$, and then $q > q$ is chosen and rewritten to $\top$.

Finally, the classical formula resulting from the normal forming is $p \rightarrow \top$, i.e. $\top$, and the decision procedure returns *valid*.

Example Let $A = p \vee (\neg p \wedge q \wedge r)) > \neg q$. The hypothesis is already in DNF. In step 2.c we factor out the atom p and replace the hypothesis $((p \wedge \top) \vee (\neg p \wedge q \wedge r) \vee (\bot)) > \neg q$ by

$$(p \wedge (\top \vee \bot) > \neg q \wedge (\top \vee (q \wedge r) \vee \bot) > (\top \vee \bot \vee \neg p > \neg q)) \vee$$

$$(\neg p \wedge ((q \wedge r) \vee \bot) > \neg q \wedge (\top \vee (q \wedge r) \vee \bot) > ((q \wedge r) \vee \bot \vee p > \neg q))$$

After eliminating $\top$ and $\bot$ by classical equivalences, we iteratedly eliminate $\top$ and $\bot$ with steps 3.a and 3.b and get

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

Then we obtain $(p \wedge \neg q) \vee (\neg p \wedge (q \wedge r) > \neg q \wedge ((q \wedge r) \vee \neg q))$ by 2.h, and finally $(p \wedge \neg q) \vee (\neg p \wedge q > r > \neg q \wedge ((q \wedge r) \vee \neg q))$ by 3.a. Now steps 3.h and 3.f apply and eliminate the remaining conditional operators. We end up with the classical formula $(p \wedge \neg q) \vee (\neg p \wedge \bot \wedge ((q \wedge r) \vee \neg q))$, which is equivalent to $p \wedge \neg q$.

5 TERMINATION

Theorem 1 (Termination) *The decision procedure terminates for every input formula F.*

Proof If there is no conditional in F then we are done.

Else a conditional $A > C$ with A and C classical is chosen and the normal forming procedure **eliminate-cond** is applied to it.

In step 1, termination of the transformations in DNF and NNF is a property of classical logic. Note that the number of atoms remains the same.

In step 2, all the hypotheses of the conditionals coming from a rewrite step are less complex than the original hypothesis (contain strictly less atoms). Precisely, to each formula F we associate a sequence $\{c_i\}_{i\in\mathbb{N}}$ such that c_i is the number of conditionals $A' > C'$ in F with $\mathrm{card}(atm(A')) = i$. Under the lexigraphic ordering this parameter decreases by every rewrite rule of step 2. Note that due to Lemma 2, step 2 is applicable whenever there is a conditional $A' > C'$ with $\mathrm{card}(atm(A')) \geq 1$. Note also that factoring out does not increase the number of atoms.

In step 3, after the application of a rewrite rule the hypotheses have the same complexity as before and the consequences of the conditonals are less complex (in particular when $L > (B \wedge C)$ and $L > (B \vee C)$ are replaced). Precisely, to each formula F we associate a sequence $\{c_i\}_{i\in\mathbb{N}}$ such that c_i is the number of conditionals $A' > C'$ in F with i symbols in C'. Under the lexicographic ordering this parameter decreases by every rewrite rule of step 3.

6 CORRECTNESS

Correctness of both the normal forming algorithm and the decision procedure follows from the fact that every replacement in the normal forming algorithm corresponds to a valid equivalence of the PMA.

First we tackle the simple things.

6.1 Correctness of the Rewrite Steps 3.a – n, 2.a, 2.b

Lemma 3 *The following equivalences are valid in the PMA.*

1. $\top > C \leftrightarrow C$
2. $\bot > C \leftrightarrow \top$
3. $L > (B \wedge C). \leftrightarrow .L > B \wedge L > C$
4. $L > (B \vee C) \leftrightarrow L > B \vee L > C$
5. $L > L \leftrightarrow \top$
6. $L > \neg L \leftrightarrow \bot$
7. $\neg L > L \leftrightarrow \bot$
8. $L > L' \leftrightarrow L'$ *if* $atm(L) \neq atm(L')$
9. $A > \top \leftrightarrow \top$
10. $L > \bot \leftrightarrow \bot$
11. $(L_1 \wedge \ldots \wedge L_n) > C \leftrightarrow L_1 > (\ldots > (L_n > C)\ldots)$ *if* $L_1 \wedge \ldots \wedge L_n$ *is consistent*
12. $(L_1 \vee \ldots \vee L_n) > C \leftrightarrow$

    $((L_1 \vee \ldots \vee L_n) \wedge C) \vee (L_1 > C \wedge \ldots \wedge L_n > C)$

    if $L_1 \vee \ldots \vee L_n$ *is non-tautological*

Proof We only prove the validity of the last equivalence, the others are similar.

Let $w \subseteq ATM$ be any classical interpretation, and let $L_1 \vee \ldots \vee L_n$ be non-tautological, i.e. there are no L_i, L_j such that $L_i = \neg L_j$.

There are two cases:

i. $w \in |L_1 \vee \ldots \vee L_n|$.

Then $w \star |L_1 \vee \ldots \vee L_n| = \{w\}$.

Now suppose $w \in |(L_1 \vee \ldots \vee L_n) > C|$. By the truth condition, $w \in |C|$ as well. Hence $w \in |(L_1 \vee \ldots \vee L_n) \wedge C|$ and as well

$$w \in |((L_1 \vee \ldots \vee L_n) \wedge C) \vee (L_1 > C \wedge \ldots \wedge L_n > C)|$$

The other way round, suppose

$$w \in |((L_1 \vee \ldots \vee L_n) \wedge C) \vee (L_1 > C \wedge \ldots \wedge L_n > C)|$$

If $w \in |(L_1 \vee \ldots \vee L_n) \wedge C|$ then $w \in |C|$ and we are done. Else we must have $w \in |L_i > C|$ for every L_i. As $w \in |L_1 \vee \ldots \vee L_n|$, there must be some L_j such that $w \in |L_j|$. Then by the truth condition, $w \in |C|$ as well, and consequently $w \in |(L_1 \vee \ldots \vee L_n) > C|$.

ii. $w \notin |L_1 \vee \ldots \vee L_n|$.

Then as there are no L_i, L_j such that $L_i = \neg L_j$, it follows from the definition of a distance that $w \star |L_1 \vee \ldots \vee L_n| = w \star |L_1| \cup \ldots \cup w \star |L_n|$.

Suppose $w \star |L_1 \vee \ldots \vee L_n| \subseteq |C|$. This means that $w \star |L_i| \subseteq |C|$ for every L_i. Hence $w \in |L_1 > C \wedge \ldots \wedge L_n > C|$ and as well

$$w \in |((L_1 \vee \ldots \vee L_n) \wedge C) \vee (L_1 > C \wedge \ldots \wedge L_n > C)|$$

The other way round, suppose

$w \in |((L_1 \vee \ldots \vee L_n) \wedge C) \vee (L_1 > C \wedge \ldots \wedge L_n > C)|$

As $w \notin |L_1 \vee \ldots \vee L_n|$, we must have $w \in |L_i > C|$ for every L_i. The truth condition warrants that $w\star|L_i| \subseteq |C|$ for every L_i. As we have said above, we also have $w\star|L_1 \vee \ldots \vee L_n| \subseteq |C|$, which means $w \in |(L_1 \vee \ldots \vee L_n) > C|$.

6.2 Some Semantical Properties

Before establishing the correctness of step 2.c, we first prove several more involved properties of the PMA semantics. They concern the interaction between the notion of minimal change of the PMA and the classical connectives.

Proposition 2 *Let $v \in w\star|A|$.*

1. *$w \in |A|$ iff $w\dot{-}v = \emptyset$ iff $w\star|A| = \{w\}$.*
2. *$w\dot{-}v \subseteq atm(A)$.*
3. *If $w \notin |A|$ then $(w\dot{-}v) \cap atm(A) \neq \emptyset$.*
4. *For every B such that A and B do not interfere, $(w\dot{-}v) \cap atm(B) = \emptyset$.*

Proof
(1.) follows from the definition of distance.
To prove (2.) suppose the contrary: then the distance between w and v would not be minimal (contradicting that $v \notin w\star|A|$).
(3.) follows from (1), and (4.) follows from (2).

Proposition 3 *Suppose A and B do not interfere.*

1. *If $w \in |A|$ then $w\star|A \wedge B| = w\star|B|$.*
2. *If $w \notin |A|$ then $w\star|B| \subseteq w\star|(A \wedge B') \vee B|$.*
3. *$w\star|A \wedge B| = (w\star|A|)\star|B|$*

Proof

1. First, $w\star|A\wedge B| = w\star|A|\star|B|$ by (3) of Proposition 3. By (1) of Proposition 2, $w\star|A| = \{w\}$, hence $w\star|A|\star|B| = \{w\}\star|B| = w\star|B|$.
2. Suppose $v \in w\star|B|$. We prove that there is no $v' \in w\star|A \wedge B'|$ closer to w than v, establishing thus that $v \in w\star|(A \wedge B') \vee B|$. As $w \notin |A|$, we must have $(w\dot{-}v') \cap atm(A) \neq \emptyset$ by (3.) of Proposition 2. On the other hand, as A and B do not interfere, $(w\dot{-}v) \cap atm(A) = \emptyset$. Hence we can never have $w\dot{-}v' \subset w\dot{-}v$, and therefore $v \in w\star|(A \wedge B') \vee B|$.
3. This follows from (4.) of Proposition 2.

Proposition 4 *Suppose A and B do not interfere, and $w \in |A|$. Then*

$$w\star|(A \wedge B) \vee B'| = w\star|B \vee B'|$$

Proof $w\star|(A \wedge B) \vee B'|$ is the set of those elements of $w\star|A\wedge B| \cup w\star|B'|$ whose distance to w is minimal. (To see that, suppose $v \in w\star|B|$ and $v \notin w\star|(A\wedge B)\vee B'|$. The latter means that there is a $v' \in w\star|A \wedge B|$ such that $w\dot{-}v' \subset w\dot{-}v$.) As $w \in |A|$ and A and B do not interfere, by (1.) of Proposition 3 this set is equal to the minimal elements of $w\star|B| \cup w\star|B'|$. But the latter is nothing else than $w\star|B \vee B'|$.

Proposition 5 *Suppose A and B as well as A and B' do not interfere, and $w \notin |A|$. Then*

$$w\star|(A\wedge B)\vee B'| = (((w\star|B\vee B'|) - |B'|)\star|A|) \cup (w\star|B'|)$$

Proof From the right to the left: If $v \in w\star|B'|$, then as $w \notin |A|$ and A and B' do not interfere, $v \in w\star|(A \wedge B) \vee B'|$ follows from (6) of Proposition 3. Hence suppose $v \in (w\star|B \vee B'| - |B'|)\star|A|$. This means that there is some $v_B \in w\star|B \vee B'|$ such that $v \in v_B\star|A|$ and $v_B \notin |B'|$. Thus $v_B \in w\star|B|$, and there is no $v_{B'} \in v\star|B'|$ closer to w. In other words, for every $v_{B'}$ we have $w\dot{-}v_{B'} \subseteq atm(B')$, and there is some atom $p \in w\dot{-}v_{B'}$ such that $p \notin w\dot{-}v_B$. Hence (and this is the crucial step) as A and B' do not interfere, by (4) of Proposition 2 we have also that $p \notin w\dot{-}v$. This means $v \in w\star|B|\star|A|$, and moreover there is no $v_{B'} \in v\star|B'|$ such that $w\dot{-}v_{B'} \subset w\dot{-}v$. As A and B do not interfere, by (3) of Proposition 3 we have $v \in w\star|A \wedge B|$, and thus $v \in w\star|(A \wedge B) \vee B'|$ because there is no $v_{B'} \in v\star|B'|$ closer to w.

From the left to the right: Suppose $v \in w\star|(A \wedge B) \vee B'|$. Then either $v \in w\star|B'|$ and we are done, or $v \in w\star|A \wedge B|$, and there is no $v' \in w\star|B'|$ which is closer to w than v. In the latter case $v \in w\star|B|\star|A|$ because A and B do not interfere. This means that there is some $v_B \in w\star|B \vee B'|$ such that $v \in v_B\star|A|$ and $v_B \notin |B'|$. Hence $v \in (w\star|B \vee B'| - |B'|)\star|A|$.

6.3 Correctness of Rewrite Step 3.c

Proposition 6 *Suppose A and B, A and B' and A and B'' do not interfere. Then the following formulas are valid in the PMA.*

1. $A \rightarrow .((A\wedge B)\vee B') > C \leftrightarrow (B \vee B') > C$
2. $\neg A \rightarrow .((A\wedge B)\vee B') > C \leftrightarrow (B' > C \wedge (B \vee B') > (B' \vee (A > C)))$

3. $((A \wedge B) \vee (\neg A \wedge B') \vee B'') > C \leftrightarrow$.

$(A \wedge (B \vee (\neg A \wedge B') \vee B'') > C) \vee (\neg A \wedge ((A \wedge B) \vee B' \vee B'') > C$

4. $((A \wedge B) \vee B')) > C \leftrightarrow$.

$(A \wedge (B \vee B') > C) \vee (\neg A \wedge B' > C \wedge (B \vee B') > (B' \vee (A > C)))$

Proof
(1.) and (2.) are the syntactical counterpart of Proposition 4.
Let us prove (3.): First, $((A \wedge B) \vee (\neg A \wedge B') \vee B'') > C$ is equivalent to

$((A \wedge B) \vee (\neg A \wedge B') \vee B'') > C \wedge A . \vee . ((A \wedge B) \vee (\neg A \wedge B') \vee B'') > C \wedge \neg A$

Then the equivalence follows with (1.). To establish (4.), (1.) and (2.) are applied in the same manner as in the proof of (3.).

Lemma 4 *Suppose A and B, A and B', and A and B'' do not interfere. Then the equivalence*

$((A \wedge B) \vee (\neg A \wedge B') \vee B'') > C \leftrightarrow$

$(A \wedge (B \vee B'') > C \wedge (B \vee B' \vee B'') > (B \vee B'' \vee \neg A > C)) \vee$

$(\neg A \wedge (B' \vee B'') > C \wedge (B \vee B' \vee B'') > (B' \vee B'' \vee A > C))$

is valid in the PMA.

Proof Apply the equivalence (3.) of Proposition 6 and then twice (4.).

6.4 Correctness of the Decision Procedure

Theorem 2 (Correctness) *Let F be a formula. If F is valid in the PMA then the decision procedure returns* valid, *and else it returns* non-valid.

Proof First, by the Termination Theorem 1, the decision procedure terminates.

Now all the steps in the normal forming algorithm **eliminate-cond** replace formulas by equivalent ones: for step 1 this is ensured by Lemma 1. For the rewrite rules of step 2.a - n, the equivalence is warranted by Lemma 4 and Lemma 2. Steps 2.a and 2.b and step 3.a - d are covered by Lemma 3. Hence each time the normal forming algorithm is applied, it returns a classical formula that is equivalent to its input formula.

As the number of conditional operators decreases with each call of **eliminate-cond**, by iteration a classical formula is obtained that is equivalent to the original formula. Finally, the decision procedure for classical logic establishes the result.

7 COMPLEXITY

It has been proved by Eiter and Gottlob (1992) that the theoretical complexity of all the known update formalisms is strictly higher than that of classical logic. Precisely, they have shown that deciding in the PMA if a counterfactual $p > q$ is true over a database KB is Π_2^p-complete, and that Π_2^p-hardness holds even if $KB \subseteq ATM$ and q is an atom with $q \in KB$ (Eiter and Gottlob 1992, Theorem 6.4).

Clearly, our decision procedure is exponential in the size of the update formula. Nevertheless, it is only co-NP-complete in the size of the database (the size of the update formula p being less or equal some constant p). This is exactly the worst case complexity of the PMA given by Eiter and Gottlob (1992) in their Theorem 8.1 (see also their Table 3).

A considerably simpler normal form can be obtained when the hypotheses are restricted to clauses or conjunctions of literals. Thus we improve considerably the complexity bound for the general case as given in (Eiter and Gottlob 1992), answering at the same time an open question from the latter.

In the case of conjunctions of literals, normal forming can be done in linear time:

Theorem 3 *If every hypothesis of a formula F is a cube, then normal forming is in $O(n)$, where n is the number of symbols in F.*

The case of clauses is slightly more complex than that of cubes. Here normal forming can be done in quadratic time:

Theorem 4 *If every hypothesis of a formula F is a clause or a cube, then normal forming is in $O(n^2)$, where n is the number of symbols in F.*

As a corollary we get that in this case the complexity of the decision problem for the PMA is the same as that for classical logic.

Corollary 1 *For the class of formulas where every hypothesis is a clause or a cube, the validity problem is co-NP-complete.*

8 UPDATES UNDER INTEGRITY CONSTRAINTS

One story to learn from Lifschitz's counterexample of section 2.2 is that there are aspects of the domain structure which cannot be expressed by classical integrity constraints. In (Fariñas and Herzig 1994a,

1994b, 1996) (see also (Giunchiglia 1995)), we have argued for the use of information about dependences between formulas in change operations, and we have studied the formal properties of the appropriate dependence notion.

Dependence is a sort of weak causal connection. Our reading of 'q depends on p' is that updates concerning p *may* change the truth value of q. In the example, l depends on up_1, but up_2 does not depend on up_1: for the update of any database by up_1 or $\neg up_1$, the truth value of up_2 remains unchanged.

Formally, we suppose given a *dependence function* DEP mapping atoms to sets of atoms such that $p \in DEP(p)$ for all atoms p. We shall suppose in the sequel that $DEP(p)$ is *finite* for every p. E.g. $DEP(up_1) = \{l\}$, $DEP(up_2) = \{l\}$, and $DEP(l) = \{up_1, up_2\}$. Dependence can be extended to general formulas by stipulating $DEP(A) = \bigcup_{p \in atm(A)} DEP(p)$.

Now we must define an update operation $\star_{IC,DEP}$ such that for every update A:

- atoms that are independent of A 'survive': if $p \notin DEP(A)$ and $KB \models p$ then we want $KB \star_{IC,DEP} A \models p$, and
- atoms that depend on A are *a priori* retracted: if $p \in DEP(A)$ then we want $KB \star_{IC,DEP} A \not\models p$ and $KB \star_{IC,DEP} A \not\models \neg p$.

E.g. in our example we want $KB \star_{IC,DEP} \neg up_1 \models up_2$ because $up_2 \notin DEP(up_1)$. On the other hand, *a priori* neither l nor $\neg l$ will follow from $KB \star_{IC,DEP} \neg up_1$ because $l \in DEP(up_1)$. Now a way to obtain that l follows from the updated database is to simply add IC to it.

Therefore, we define an update operation $\star_{IC,DEP}$ under a set of integrity constraints and a dependence function DEP as follows:

$$KB \star_{IC,DEP} A = (\bigvee_{B \in CTX(A)} (KB \star B)) \wedge IC \wedge A$$

where $\star$ is the PMA update operation, and the function CTX is defined by

$$CTX(A) = \{L_1 \wedge \ldots \wedge L_n : L_i = p_i \text{ or } L_i = \neg p_i\}$$

if $DEP(A) = \{p_1, \ldots, p_n\}$. E.g. $CTX(up_1) = \{up_1 \wedge l, up_1 \wedge \neg l, \neg up_1 \wedge l, \neg up_1 \wedge \neg l\}$. Equivalently, we have

$$KB \star_{IC,DEP} A = \bigvee_{B \in CTX(A)} ((KB \star B) \wedge IC \wedge A)$$

We reformulate with conditionals and get

$KB \star_{IC,DEP} A \models C$

iff $\bigvee_{B \in CTX(A)} (KB \star B) \wedge IC \wedge A \models C$

iff $KB \star B \models (IC \wedge A) \rightarrow C$
for every $B \in CTX(A)$

iff $KB \rightarrow (B > ((IC \wedge A) \rightarrow C)$
for every $B \in CTX(A)$

In this way we have defined a new update operation, and a first thing to do is to relate it to the PMA: suppose $IC = \emptyset$ and $DEP(p) = \{p\}$ for all $p \in ATM$. It turns out that $KB \star_{IC,DEP} A = KB \star A$ for every A that is a cube. The behaviour is different if A is a clause: e.g. $\{\neg p, \neg q\} \star_{IC,DEP} (p \vee q) = \{\{p, q\}, \{p\}, \{q\}\}$, whereas we have seen in section 2.2 that $\{\neg p, \neg q\} \star (p \vee q) = \{\{p\}, \{q\}\}$. Note that (as we have already mentioned there) the PMA has been criticized for that exclusive interpretation of the inclusive or.

In our example, we have to check the validity of four formulas. We only give the case of $KB \rightarrow (up_1 \wedge l) > ((IC \wedge \neg up_1) \rightarrow \neg l)$.

First, the conditional $(up_1 \wedge l) > ((IC \wedge \neg up_1) \rightarrow \neg l)$ is rewritten in step 2.a of the normal forming algorithm to $up_1 > l > ((IC \wedge \neg up_1) \rightarrow \neg l)$. Then, the conditional $l > ((IC \wedge \neg up_1) \rightarrow \neg l)$ is rewritten applying several times steps 3.c and 3.d to $(l > ((up_1 \leftrightarrow up_2) \leftrightarrow l) \wedge l > \neg up_1) \rightarrow l > \neg l$. The first conditional $(l > ((up_1 \leftrightarrow up_2) \leftrightarrow l)$ is rewritten in several steps to $(up_1 \leftrightarrow up_2)$. [2] The second one $l > (\neg up_1)$ is rewritten to $\neg up_1$, and the third one $l > \neg l$ to $\bot$. Finally we obtain the classical formula $((up_1 \leftrightarrow up_2) \wedge \neg up_1) \rightarrow \bot$, which is equivalent to $up_1 \vee up_2$.

Now we still have to rewrite $l > (up_1 \vee up_2)$ to $up_1 \vee up_2$, and to check whether $KB \rightarrow (up_1 \vee up_2)$ is classically valid, which is the case.

Similarly, the three other updates are performed. It turns out that $KB \star_{IC,DEP} \models l$, i.e. our approach treats the counterexample correctly.

Generally, if $\text{card}(DEP(A)) = n$ we must do 2^n PMA-updates. Clearly, such a method is interesting only if knowledge is represented in a way such that there are very few dependences. On the other hand, a very advantageous feature of our approach is that updates are limited to conjunctions of literals. As we have seen in Theorem 3, normal forming can be done in linear time in this case. Therefore, if we suppose that there is

[2] Remember that we have kept $\leftrightarrow$ and $\rightarrow$ here for reasons of readability. In the algorithm, it is supposed that only $\wedge, \vee$ and $\neg$ appear in formulas.

some constant k such that $\text{card}(DEP(p)) \leq k$ for all $p \in ATM$, (which is the case in particular if our language is finite) then the complexity of deciding updates under integrity constraints and dependences becomes that of deciding the validity of a classical formula.

9 DISCUSSION

Although the theoretical complexity of the validity problem in the PMA is strictly higher than that for classical logic, in practical applications updates are often conjunctions or disjunctions of literals, for which we have given better complexity results.

In particular, we have shown how integrity constraints and dependences can be represented in our approach in a way such that the only updates involved are conjunctions of literals. We have pointed out that if moreover the dependences are bounded then the complexity of the decision problem for the PMA is the same as that for classical logic.

In a conditional language the updates are hypothetical and the database is not progressed (in Reiter's sense). It is an open problem how the updated database can be effectively constructed. Another open problem is whether our method can be extended to other approaches such as Dalal's (1988).

Acknowledgements

This paper continues work done with Luis Fariñas del Cerro that has been presented in (Fariñas and Herzig 1988, 1994c, 1997). Thanks to him as well as to Bernhard Nebel for several stimulating discussions.

References

J. P. Burgess (1981), Quick completeness proofs for some logics of conditionals. Notre Dame J. of Formal Logic 22 1, pp 76-84.

T.S. Chou and M. Winslett (1991), Immortal: A model-based belief revision system. Proc. Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'91), Morgan Kaufmann.

T.S. Chou and M. Winslett (1994), A model-based belief revision system. J. of Automated Reasoning 12, pp 157-208.

M. Dalal (1988), Investigations into a theory of knowledge base revision: preliminary report. Proc. 7th Nat. Conf. on Artificial Intelligence (AAAI'88).

A. Del Val (1992), Computing knowledge base updates. Proc. Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'92), Morgan Kaufmann.

Th. Eiter and G. Gottlob (1992), On the complexity of propositional knowledge base revision, updates, and counterfactuals. Journal of Artificial Intelligence 57, pp 227-270.

Th. Eiter, G. Gottlob, and Y. Gurevich (1993), Curb your theory! A circumscriptive approach for inclusive interpretation of disjunctive information. Proc. IJCAI'93.

L. Fariñas del Cerro and A. Herzig (1988), An automated modal logic for elementary changes. In: P Smets, A. Mamdani, D. Dubois and H. Prade (eds.), Non-Standard Logics for Automated Reasoning, Academic Press, pp 63-79.

L. Fariñas del Cerro and A. Herzig (1994a), Interference logic = conditional logic + frame axiom. Int. J. of Intelligent Systems, Special Issue on Revision and Updating in Knowledge Bases, 9(1), pp 119-130.

L. Fariñas del Cerro, A. Herzig (1994b), Possibility theory and independence. In: B. Bouchon-Meunier, R.R. Yager, L. A. Zadeh, eds., Advances in Intelligent Computing - IPMU-94. Springer Verlag, LNCS 945, pp 292-301.

L. Fariñas del Cerro et A. Herzig (1994c), A conditional logic for updating in the Possible Models Approach. Proc. 18th German Conf. on Artificial Intelligence (KI'94), eds. B. Nebel, L. Dreschler-Fischer, Springer Verlag, LNAI 861.

L. Fariñas and A. Herzig (1996), Belief change and dependence. Proc. 6th Conf. on Theoretical Aspects of rationality and knowledge (TARK VI), March 1996.

L. Fariñas and A. Herzig (1997), All other things being equal: on a notion of inertia in conditional logic", In: E. Ejerhed and S. Lindström (eds.), Logic, Action and Cognition - Selection of papers of the Umea colloquium on Dynamic approaches in Logic, Language and Information (UmLLI-93). KluwerAcademic Publishers, to appear. (http://www.irit.fr/ACTIVITES/EQ_ALG/Papers/umea.ps)

P. Gärdenfors (1988), Knowledge in Flux. MIT Press.

E. Giunchiglia and V. Lifschitz (1995), Dependent fluents. Proc. IJCAI'95, Montreal, Canada, pp 1964-1969.

G. Grahne (1991), Updates and counterfactuals. Proc.

Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'91), Morgan Kaufmann.

G. Grahne and A. O. Mendelzon (1995), Updates and subjunctive queries. Information and Computation 116, pp 241-252.

H. Katsuno and A.O. Mendelzon (1991), On the Difference between Updating a Knowledge Base and Revising it. In: P. Gärdenfors (ed.), Belief Revision. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, pp 183-203. Preliminary version in: Proc. Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'91), Morgan Kaufmann.

V. Lifschitz (1986), Frames in the space of situations. Journal of Artificial Intelligence 46, pp 365-376.

M. D. Ryan and P.-Y. Schobbens (1997), Intertranslating Counterfactuals and Updates. In: J. of Logic, Language and Information, special issue on TARK'96, to appear. (Preliminary version in: W. Wahlster (ed.), Proc. ECAI'96. John Wiley & Sons, Ltd., to appear.)

M. Winslett (1988), Reasoning about actions. Proc. 7th Nat. Conf. on Artificial Intelligence (AAAI'88).

M. Winslett (1990), Updating Logical Databases. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press.

M. Winslett (1995), 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", Oxford University Press, pp.133-174.

Causality and the Qualification Problem

Michael Thielscher*
International Computer Science Institute
1947 Center Street
Berkeley, CA 94704-1198

Abstract

In formal theories for reasoning about actions, the qualification problem denotes the problem to account for the many conditions which, albeit being unlikely to occur, may prevent the successful execution of an action. By a simple counter-example in the spirit of the well-known Yale Shooting scenario, we show that the common straightforward approach of globally minimizing such *abnormal disqualifications* is inadequate as it lacks an appropriate notion of causality. To overcome this difficulty, we propose to incorporate causality by treating the proposition that an action is qualified as a *fluent* which is initially assumed away by default but otherwise potentially indirectly affected by the execution of actions. Our formal account of the qualification problem includes the proliferation of explanations for surprising disqualifications and also accommodates so-called miraculous disqualifications. We moreover sketch a version of the fluent calculus which involves default rules to address abnormal disqualifications of actions, and which is provably correct wrt. our formal characterization of the qualification problem.

1 INTRODUCTION

A fundamental requirement for autonomous intelligent agents is the ability to reason about causality, which enables the agent to understand the world to an extent sufficient for acting intelligently on the basis of his or her knowledge as to the effects of actions. The *qualification problem* [McCarthy, 1977] in formal theories for reasoning about actions arises from the fact that generally the successful execution of actions depends on many more conditions than we are usually aware of. The reason for this unawareness is that most conditions are so likely to be satisfied that they are assumed away in case there is no evidence to the contrary.

A standard example to illustrate this is when we intend to start our car's engine, then we usually do not make sure that no potato in the tail pipe prevents us from doing so, despite the fact that a clogged tail pipe necessarily renders this action impossible.[1] While this *prima facie* ignorance is rational as it is generally impossible to verify all possible preconditions,[2] they cannot be completely disregarded in a sound formal model. Yet a proposition like "there is no potato in the tail pipe" should not be treated as a *strict* precondition in the formal specification of the action "start the engine" lest the reasoning agent always has to verify this condition before assuming that the action can be successfully executed. Moreover, it is often difficult if not impossible to even think of all conceivable disqualifications in advance [McCarthy, 1977].

Allowing to assume away all so-called *abnormal* disqualifications by default naturally implies that if further knowledge hints at any such unexpected disqualification, then the previous conclusion that the action in question be qualified needs to be withdrawn. Thus the entire process is intrinsically nonmonotonic. As a consequence, McCarthy's proposal was to employ circumscription with the aim of minimizing abnormal disqualifications [McCarthy, 1977; McCarthy, 1980; McCarthy, 1986]. Little has been achieved since then towards formally integrating this concept into a specific action formalism, or towards an assessment of its range of applicability. In fact, a surprisingly simple example illustrates that the straightforward global minimization of abnormal disqualifications is inadequate. The example shows some similarities to the problem—

*On leave from FG Intellektik, TH Darmstadt.

[1] According to [Ginsberg and Smith, 1988b], this example is due to McCarthy.

[2] Aside from the fact that besides a clear tail pipe there are lots of other disqualifying, albeit unlikely, obstacles, how can we ensure that after checking the tail pipe it does not become clogged during us walking to the front door and taking a seat, prior to trying to start the engine?

first illustrated with the Yale Shooting domain [Hanks and McDermott, 1987]—which occurs when neglecting causality in tackling the frame problem.

Imagine the following scenario: We can put a potato into the tail pipe whenever no abnormal disqualification prevents us from doing so (e.g., the potato surprisingly turns out to be too heavy); likewise we can start the engine except in case of an abnormal disqualification (like a potato in the tail pipe). Now, what would we predict as to the outcome of first trying to place a potato in the tail pipe and, then, trying to start the engine? Clearly, since nothing hints at an abnormal disqualification of the first action, we should expect this one to be successful. Then its effect (viz. a potato in the tail pipe) implies that the second action will be unqualified.

But what happens if abnormal disqualifications are globally minimized in this scenario? One minimal model is obviously obtained by considering the *put-potato* action qualified and the *start-engine* action unqualified, as expected. However, if instead the first action is assumed unqualified, then this in turn avoids the necessity of assuming a disqualification of the second. For if *put-potato* is not qualified, then it fails to produce what otherwise causes the disqualification of *start-engine*. Hence, in so doing we can construct a second minimal model for our scenario—which is clearly unintended.

The reason for the existence of the second, counterintuitive model is that global minimization does not allow to distinguish disqualifications which can be explained from the standpoint of causality. Successfully introducing a potato into the tail pipe produces an effect which *causes* the fact that the second action, starting the engine, is unqualified. That is to say, while an abnormal disqualification of *put-potato* comes out of the blue in the unintended minimal model, an abnormal disqualification of *start-engine*, as claimed in the first minimal model, is easily explicable. One even tends to not call abnormal this situation since being unable to start the engine after having clogged the tail pipe is, after all, what one would normally expect. The reader might notice the similarities to the Yale Shooting problem: A gun that becomes magically unloaded while waiting deserves being called abnormal, whereas causality explains the death of the turkey if being shot at with a loaded gun [Hanks and McDermott, 1987].

The only existing alternative to global minimization of abnormalities as an approach to the qualification problem is based on *chronological ignorance* [Shoham, 1987; Shoham, 1988]. The basic idea there is to assume away by default abnormal, disqualifying circumstances, and simultaneously to prefer minimization of abnormalities at earlier timepoints. While this method treats our example scenario correctly, it is inherently incapable of handling non-deterministic actions, or non-deterministic information in general, as has already been argued elsewhere. A detailed account of this approach is given in the concluding discussion, Section 5.

Given the inadequacy of global minimization and the limited expressiveness of chronological ignorance, we propose a formal account of the qualification problem which incorporates a suitable concept of causality. We accomplish this by taking the proposition that an action is abnormally disqualified as a fluent, i.e., a proposition that may change its truth value in the course of time.[3] This proposition is assumed false, by default, *initially*, and by virtue of being fluent, it may be affected by the execution of an action and otherwise is subject to the general law of persistence. This helps to distinguish action disqualifications which are (indirectly) caused by actions that have been observed. As this method requires an appropriate treatment of indirect effects, we will adopt the approach to the *ramification problem* proposed in [Thielscher, 1997], where indirect effects are obtained according to so-called causal relationships among fluents. As a side gain, this enables us to account for *implicit* strict preconditions of actions, which are not part of an action specification but derive from certain domain constraints. This is sometimes considered part of the qualification problem, e.g. in [Ginsberg and Smith, 1988b; Lin and Reiter, 1994].

Aside from providing means to assume away abnormal disqualifications by default while properly taking into account possible causes for these disqualifications, the successful treatment of the qualification problem should include the proliferation of explanations in case an action has been—unexpectedly—observed unqualified. It may of course happen, though, that we are still unable to perform an action even if we have explicitly excluded, to the best of our knowledge, any imaginable preventing cause. However surprising this might be, it just shows us that we have only partial knowledge of the world. We call *miraculous* a disqualification which cannot be explained even if abnormal circumstances are granted. Consequently, miraculous disqualifications are to be minimized with higher priority than abnormal disqualifications which admit an explanation. Another characteristics of miraculous disqualifications is that they may occur or vanish even if, from our perspective, the situation has not changed. Again this is due to our lack of omniscience. The formal account of the qualification problem presented in this paper addresses both finding explanations for unexpectedly observed disqualifications and accounting for miraculous disqualifications.

We moreover sketch, on the basis of the *fluent calculus* [Hölldobler and Schneeberger, 1990; Thielscher,

[3] Throughout the paper, by "(dis-)qualified" we mean "physically (im-)possible." The refinement that actions may be unqualified *as to producing a certain effect* will be discussed at the end, in Section 5.

1997], an action calculus which includes a proper treatment of abnormal disqualifications. Since the qualification problem requires some sort of nonmonotonic feature, we employ *default rules* in the sense of [Reiter, 1980] to formalize the initial normality assumptions as well as the assumption that miraculous disqualifications do not occur. The resulting action calculus is provably correct wrt. our formal characterization of the qualification problem.

2 ACTIONS AND RAMIFICATIONS

2.1 A BASIC THEORY OF ACTIONS

The basic entities of action scenarios are *states*, each of which is a snapshot of the underlying dynamic system, i.e., the part of the world being modeled, at a particular instant. Formally, a state is determined by an assignment of truth values to a fixed set of propositional constants.[4]

Definition 1 Let $\mathcal{F}$ be a finite set of symbols called *fluent names*. A *fluent literal* is either a fluent name $f \in \mathcal{F}$ or its negation, denoted by $\overline{f}$. A set of fluent literals is *inconsistent* iff it contains some $f \in \mathcal{F}$ along with $\overline{f}$. A *state* is a maximal consistent set of fluent literals. ■

Notice that formally any combination of truth values denotes a state, which, however, might be considered impossible due to specific dependencies among the fluents (see below). Throughout the paper we assume the following notational conventions: If ℓ is a fluent literal, then $|\ell|$ denotes its affirmative component, that is, $|f| = |\overline{f}| = f$ where $f \in \mathcal{F}$. This notation extends to sets of fluent literals S as follows: $|S| = \{|\ell| : \ell \in S\}$. E.g., for each state S we have $|S| = \mathcal{F}$. Furthermore, if $\ell = \overline{f}$ is a negative fluent literal then $\overline{\ell}$ should be interpreted as f.

The elements of an underlying set of fluent names can be considered atoms for constructing (propositional) formulas to allow for statements about states. Each fluent literal and $\top$ (*tautology*) and $\bot$ (*contradiction*) are *fluent formulas*, and if F and G are fluent formulas then so are $F \wedge G$, $F \vee G$, $F \supset G$, and $F \equiv G$.[5] The notion of fluent formulas being *true* in a state S is based on defining a literal ℓ to be true if and only if $\ell \in S$. Fluent formulas provide means to distinguish states that cannot occur due to specific dependencies among particular fluents. Formulas which have to be satisfied in all states that are possible in a domain are also called *domain constraints*.

[4] The calculus described in Section 4 employs a more expressive language, which involves non-propositional fluents.

[5] As negation can be expressed through negative literals, we omit the standard connective "$\neg$". This is just for the sake of readability as it avoids too many different forms of negation.

Example 1 A basic version of the Potato In Tail Pipe scenario shall be formalized with the fluent names $\mathcal{F} = \{pot, clog, runs, heavy\}$ to state whether, respectively, there is a potato in the tail pipe, the tail pipe is clogged, the engine is running, and the potato is too heavy. The fluent formula

$$pot \supset clog \tag{1}$$

then expresses the fact that the tail pipe is clogged whenever it houses a potato. Taken as domain constraint, this formula is true, for instance, in the state $\{\overline{pot}, \overline{clog}, \overline{runs}, heavy\}$. ■

The second basic entity in theories of actions are the *actions* themselves, whose execution causes state transitions. Since stress shall lie on the qualification problem rather than on sophisticated methods of specifying the direct effects of actions, we employ a suitably simple, STRIPS-style [Fikes and Nilsson, 1971] notion of action specification. Each *action law* consists of

- A *condition* C, which is a set of fluent literals all of which must be contained in the state at hand in order to apply the action law.
- A (direct) *effect* E, which is a set of fluent literals, too, all of which hold in the resulting state after having applied the action law.

It is assumed that $|C| = |E|$, that is, condition and effect refer to the very same set of fluent names. This is just for the sake of simplicity, for it enables us to obtain the state resulting from the direct effect by simply removing set C from the state at hand and adding set E to it. This assumption does not impose a restriction of expressiveness since we allow several laws for a single action, and since any (unrestricted) action law can be replaced by an equivalent set of action laws which obey the assumption.

Definition 2 Let $\mathcal{F}$ be a set of fluent names, and let $\mathcal{A}$ be a finite set of symbols, called *action names*, such that $\mathcal{F} \cap \mathcal{A} = \{\}$. An *action law* is a triple $\langle C, a, E \rangle$ where C and E are consistent sets of fluent literals such that $|C| = |E|$, and $a \in \mathcal{A}$.

If S is a state, then an action law $\alpha = \langle C, a, E \rangle$ is *applicable* in S iff $C \subseteq S$. The *application* of α to S yields the state $(S \setminus C) \cup E$. ■

Obviously, S being a state, C and E being consistent, and $|C| = |E|$ guarantee $(S \setminus C) \cup E$ to be a state again—not necessarily, however, one which satisfies the underlying domain constraints.

Example 1 (continued) We define the action names *start* (starting the engine) and *put-p* (putting a potato into the tail pipe), which are accompanied by these action laws:

$$\begin{array}{c} \langle \{\overline{runs}\}, start, \{runs\} \rangle \\ \langle \{\overline{pot}\}, put\text{-}p, \{pot\} \rangle \end{array} \tag{2}$$

In words, starting the engine is possible if it is not running and causes it to do so; similarly, a potato may be added to the tail pipe. The second law, say, is applicable in the state $S = \{\overline{pot}, \overline{clog}, \overline{runs}, \overline{heavy}\}$ since $\{\overline{pot}\} \subseteq S$. Its application yields $(S \setminus \{\overline{pot}\}) \cup \{pot\}$, i.e., $\{pot, \overline{clog}, \overline{runs}, \overline{heavy}\}$, which constitutes a state but does not satisfy our constraint, $pot \supset clog$. ■

The example illustrates that a state obtained through the application of an action law may violate the underlying domain constraints since only direct effects have been specified: Putting a potato into the tail pipe has the *indirect* effect that the latter becomes clogged. The problem of accommodating additional, indirect effects is commonly referred to as the *ramification problem* [Ginsberg and Smith, 1988a]. Prior to discussing a suitable solution, observe that according to Definition 2 it is possible to construct a set of action laws which, given a state, contains more than one applicable law for a single action name. This can be used to formalize non-deterministic actions.

Example 2 Suppose we park our car in a neighborhood that is known for its suffering from a tail pipe marauder.[6] We therefore must expect that after waiting for a certain amount of time, a potato may have randomly been introduced into our car's tail pipe. This is formally captured by giving a non-deterministic specification of an action with the name *wait*. Let $\mathcal{F} = \{pot, clog, runs\}$ and $\mathcal{A} = \{wait, start\}$. Performing a *wait* action either has no effect at all, or else it causes *pot* become true provided there is not already a potato in the tail pipe. Accordingly, we employ the following two action laws:

$$\langle \{\}, wait, \{\} \rangle \quad \text{and} \quad \langle \{\overline{pot}\}, wait, \{pot\} \rangle \qquad (3)$$

Both of them are applicable, for instance, in the state $\{\overline{pot}, \overline{clog}, \overline{runs}\}$, which suggests two possible outcomes, viz. $\{\overline{pot}, \overline{clog}, \overline{runs}\}$ and $\{pot, \overline{clog}, \overline{runs}\}$. ■

2.2 THE RAMIFICATION PROBLEM

In [Thielscher, 1997] we propose to address the ramification problem by regarding the collection of fluent literals resulting from the computation of the *direct* effects merely as an intermediate state, which requires additional computation accounting for possible *indirect* effects. More specifically, a single indirect effect is obtained according to a directed *causal* relation between two particular fluents.

Definition 3 Let $\mathcal{F}$ be a set of fluent names. A *causal relationship* is an expression of the form ε **causes** ϱ **if** Φ where Φ is a fluent formula and ε and ϱ are fluent literals. ■

[6] This example has been suggested by Erik Sandewall (personal communication).

The intended reading is the following: Under condition Φ, the (previously obtained, direct or indirect) effect ε triggers the indirect effect ϱ. E.g., the causal relationship *pot* **causes** *clog* **if** $\top$ will be used below to state that the effect *pot* always gives rise to the additional effect *clog*. Causal relationships operate on pairs (S, E), where S denotes the current state and E contains all direct and indirect effects computed so far:

Definition 4 Let (S, E) be a pair consisting of a state S and a set of fluent literals E, then a causal relationship ε **causes** ϱ **if** Φ is *applicable* to (S, E) iff $\Phi \wedge \overline{\varrho}$ is true in S and $\varepsilon \in E$. Its application yields the pair (S', E') where $S' = (S \setminus \{\overline{\varrho}\}) \cup \{\varrho\}$ and $E' = (E \setminus \{\overline{\varrho}\}) \cup \{\varrho\}$. ■

In words, 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 $(S, E) \rightsquigarrow_{\mathcal{R}} (S', E')$ we denote the existence of an element in $\mathcal{R}$ whose application to (S, E) yields (S', E'). Notice that if S is a state and E is consistent, then $(S, E) \rightsquigarrow_{\mathcal{R}} (S', E')$ implies that S' is a state and E' is consistent, too. We adopt a standard notation in writing $(S, E) \overset{*}{\rightsquigarrow}_{\mathcal{R}} (S', E')$ to indicate that there are causal relationships in $\mathcal{R}$ whose successive application to (S, E) yields (S', E').

Example 1 (continued) The following two causal relationships state respectively that the effect *pot* always gives rise to the indirect effect *clog*, and that the effect $\overline{clog}$ (as a result of clearing the tail pipe, say) always gives rise to the indirect effect $\overline{pot}$:[7]

$$\begin{array}{lll} pot & \textbf{causes} \; clog & \textbf{if} \; \top \\ \overline{clog} & \textbf{causes} \; \overline{pot} & \textbf{if} \; \top \end{array} \qquad (4)$$

Recall, now, the state $\{\overline{pot}, \overline{clog}, \overline{runs}, \overline{heavy}\}$ and action *put-p*. Applying the second action law in (2) yields the state $S = \{pot, \overline{clog}, \overline{runs}, \overline{heavy}\}$ along with the effect $E = \{pot\}$. Given the pair (S, E), the first causal relationship in (4) is applicable on account of both $\top \wedge \overline{clog}$ being true in S and $pot \in E$. The application of this relationship yields the pair $((S \setminus \{\overline{clog}\}) \cup \{clog\}, (E \setminus \{\overline{clog}\}) \cup \{clog\})$, i.e.,

$$(\{pot, clog, \overline{runs}, \overline{heavy}\}, \{pot, clog\}) \qquad (5)$$

■

Now, suppose given a set of fluent literals S as the result of having computed the direct effects of an action via Definition 2. State S may violate the domain constraints. We then compute additional, indirect effects

[7] See [Thielscher, 1997] on how a suitable set of causal relationships can be automatically extracted from domain constraints given additional knowledge as to which fluents may possibly affect each other.

by (non-deterministically) selecting and (serially) applying causal relationships. If this eventually results in a state satisfying the domain constraints, then this state is considered a *successor* state.

Definition 5 Let $\mathcal{F}$ and $\mathcal{A}$ be sets of fluent and action names, respectively, $\mathcal{L}$ a set of action laws, $\mathcal{D}$ a set of domain constraints, and $\mathcal{R}$ a set of causal relationships. Furthermore, let S be a state satisfying $\mathcal{D}$ and $a \in \mathcal{A}$. A state S' is a *successor state* of S and a iff there exists an applicable (wrt. S) action law $\langle C, a, E\rangle \in \mathcal{L}$ such that

1. $((S \setminus C) \cup E, E) \stackrel{\cdot}{\rightsquigarrow}_{\mathcal{R}} (S', E')$ for some E', and
2. S' satisfies $\mathcal{D}$. ■

Recall, for instance, the state-effect pair in (5). By virtue of satisfying our domain constraint, $pot \supset clog$, its first component constitutes a successor state of $\{\overline{pot}, \overline{clog}, \overline{runs}, \overline{heavy}\}$ and *put-p*. The analogue holds for the Tail Pipe Marauder scenario (Example 2): There are two successor states of $\{\overline{pot}, \overline{clog}, \overline{runs}\}$ and *wait*, viz. $\{\overline{pot}, \overline{clog}, \overline{runs}\}$ and $\{pot, clog, \overline{runs}\}$.

Based on Definition 5, a set of causal laws along with a set of domain constraints and a set of causal relationships determines a *causal model* Σ which maps any pair of an action name and a state to a set of states as follows: $\Sigma(a, S) := \{S' : S' \text{ successor of } S \text{ and } a\}$.

It is important to realize that neither uniqueness nor the existence of a successor state is guaranteed in general; that is, $\Sigma(a, S)$ may contain several elements or may be empty. The former characterizes actions with non-deterministic behavior even though these actions might be deterministic as regards their direct effects. If no successor exists although an applicable action law can be found, then this indicates that the action under consideration has *implicit* preconditions which are not met. While causal relationships account for these qualifications, which derive from domain constraints (see [Thielscher, 1997] for details), notice, however, that implicit preconditions still are strict and as such not part of the qualification problem dealing with the necessity of assuming away abnormal disqualifications.

3 ABNORMAL DISQUALIFICATIONS

We now take the action theory introduced in the preceding section as the basis for our formal account of the qualification problem. The general objective is to appropriately interpret a given formal scenario description and to draw reasonable conclusions about it. Any such description involves general action laws in conjunction with causal relationships, plus specific observations as to both the values of certain fluents and, especially, the non-executability of certain actions in particular situations. The term "reasonable conclusions" appeals to what common sense suggests as to how the given observations are to be interpreted. Formally, a *domain description* (or *domain*, for short) consists of sets $\mathcal{F}$ and $\mathcal{A}$ of fluent and action names; sets $\mathcal{L}$, $\mathcal{D}$, and $\mathcal{R}$ of action laws, domain constraints, and causal relationships, respectively; and a set $\mathcal{O}$ of so-called observations:

Definition 6 Let $\mathcal{F}$ and $\mathcal{A}$ be sets of fluent and action names, respectively. An *observation* is an expression of one of the following forms:

$$F \ \textbf{after} \ [a_1, \ldots, a_n] \tag{6}$$

$$a \ \textbf{disqualified after} \ [a_1, \ldots, a_n] \tag{7}$$

where F is a fluent formula and $a, a_1, \ldots, a_n$ are action names ($n \geq 0$). ■

Intuitively, observation (6) indicates that if the sequence of actions $[a_1, \ldots, a_n]$ were performed in the initial state, then F would hold in the resulting state. Likewise, (7) indicates that after performing the sequence of actions $[a_1, \ldots, a_n]$, action a would be unqualified. For instance, these are possible observations in the context of Example 1:

$$\overline{pot} \wedge \overline{runs} \ \textbf{after} \ []$$
$$start \ \textbf{disqualified after} \ [put\text{-}p]$$

In the remainder of this section, we develop formal notions of interpretations and models for domain descriptions, and we introduce a suitable preference relation among models to allow for assuming away, by default, abnormal disqualifications. This model preference criterion induces a nonmonotonic entailment relation. Together these concepts constitute our proposal how to formalize the qualification problem.

3.1 PERSISTENCE OF ACTION QUALIFICATIONS

The unintended model which occurs in the Put Potato In Tail Pipe scenario when globally minimizing abnormal disqualifications illustrates the necessity of distinguishing disqualifications that admit a *causal* explanation. We have already argued that this can be accomplished by considering the proposition whether an action is or is not abnormally disqualified as potentially being affected by the execution of other actions and otherwise being subject to the general law of persistence. In other words, this proposition is taken as a fluent. According to the general assumption that the world is 'normal' unless there is information to the contrary, this fluent is assumed *initially* false by default. Restricting the assumption of normality to the initial state enables us to consider it normal, as intended, when an action occurs whose effects suggest an action disqualification which, under general circumstances, would be abnormal. Formally, let, for each

action name a, $\mathit{disq}(a)$ be a fluent name. The intended meaning is that if $\overline{\mathit{disq}(a)}$ holds in some state, then action a is not disqualified for some abnormal reason—which shall imply that a be qualified if and only if all strict preconditions are satisfied.[8]

Abnormal disqualifications indicate abnormal circumstances. These may be described by fluents which, too, are to be assumed false by default. Example fluents of this kind might be *clog* and *pot*, as one normally assumes that the tail pipe is not clogged, let alone the possibility of its housing a potato. Fluents denoting abnormal circumstances can be combined in domain constraints to describe the conditions for an action being abnormally disqualified. In particular, it is often desirable to equate a fluent $\mathit{disq}(a)$ with a disjunction consisting of all (to the best of the agent's knowledge) the causes for an abnormal disqualification of a. This does not only allow to derive an action disqualification from the occurrence of one of its causes, it also supports the proliferation of explanations for abnormal disqualifications that have been observed (see Section 3.2.2, below).

To make all this precise, let $\mathcal{F}$ and $\mathcal{A}$ be the sets of fluent and action names, respectively, of a domain description. From now on we always assume determined a certain subset $\mathcal{F}_{ab} \subseteq \mathcal{F}$ of fluents that will be considered initially false by default. It is moreover assumed that $\mathit{disq}(a) \in \mathcal{F}_{ab}$ for each action name $a \in \mathcal{A}$. A typical domain constraint, then, is of the form

$$\mathit{disq}(a) \equiv \bigvee_{i \in I_a} f_i \tag{8}$$

for some index set I_a such that each $f_i \in \mathcal{F}_{ab}$. That is, each of the 'abnormality' fluents f_i is a potential cause of an abnormal disqualification of action a.[9] These domain constraints may give rise to indirect effects, namely, a change of the truth value of an element in the disjunction might also affect the truth value of $\mathit{disq}(a)$.

Example 1 (continued) Let the set $\mathcal{F}_{ab}$ consist of the fluents *pot*, *clog*, *heavy*, along with $\mathit{disq}(\mathit{start})$ and $\mathit{disq}(\mathit{put\text{-}p})$. Suppose further that the set of domain constraints includes

$$\begin{aligned} \mathit{disq}(\mathit{start}) &\equiv \mathit{clog} \\ \mathit{disq}(\mathit{put\text{-}p}) &\equiv \mathit{heavy} \end{aligned} \tag{9}$$

aside from $\mathit{pot} \supset \mathit{clog}$. The additional domain constraints are accompanied by these causal relationships:

$$\begin{aligned} \overline{\mathit{clog}} \ \ \mathbf{causes} \ \ \overline{\mathit{disq}(\mathit{start})} \ \ \mathbf{if} \ \top \\ \mathit{clog} \ \ \mathbf{causes} \ \ \mathit{disq}(\mathit{start}) \ \ \mathbf{if} \ \top \\ \overline{\mathit{heavy}} \ \ \mathbf{causes} \ \ \overline{\mathit{disq}(\mathit{put\text{-}p})} \ \ \mathbf{if} \ \top \\ \mathit{heavy} \ \ \mathbf{causes} \ \ \mathit{disq}(\mathit{put\text{-}p}) \ \ \mathbf{if} \ \top \end{aligned} \tag{10}$$

in conjunction with the ones shown in (4). Suppose, now, action *put-p* is performed in the state $S = \{\overline{\mathit{pot}}, \overline{\mathit{clog}}, \overline{\mathit{runs}}, \overline{\mathit{heavy}}, \overline{\mathit{disq}(\mathit{start})}, \overline{\mathit{disq}(\mathit{put\text{-}p})}\}$. The application of the corresponding action law in (2) yields the state-effect pair

$$(\{\mathit{pot}, \overline{\mathit{clog}}, \overline{\mathit{runs}}, \overline{\mathit{heavy}}, \overline{\mathit{disq}(\mathit{start})}, \overline{\mathit{disq}(\mathit{put\text{-}p})}\}, \{\mathit{pot}\})$$

The first component does not satisfy $\mathit{pot} \supset \mathit{clog}$, but we can apply the first causal relationship in (4), viz. *pot* **causes** *clog* **if** $\top$, yielding

$$(\{\mathit{pot}, \mathit{clog}, \overline{\mathit{runs}}, \overline{\mathit{heavy}}, \overline{\mathit{disq}(\mathit{start})}, \overline{\mathit{disq}(\mathit{put\text{-}p})}\}, \{\mathit{pot}, \mathit{clog}\})$$

While now the aforementioned domain constraint is satisfied, the first fluent formula in (9) is no longer so. Yet we can further apply the appropriate causal relationship in (10), viz. *clog* **causes** $\mathit{disq}(\mathit{start})$ **if** $\top$, which results in

$$(\{\mathit{pot}, \mathit{clog}, \overline{\mathit{runs}}, \overline{\mathit{heavy}}, \mathit{disq}(\mathit{start}), \overline{\mathit{disq}(\mathit{put\text{-}p})}\}, \{\mathit{pot}, \mathit{clog}, \mathit{disq}(\mathit{start})\}) \tag{11}$$

This pair's first component satisfies all domain constraints and, thus, constitutes a successor state. Notice that action *start* is declared abnormally disqualified in the resulting state. This disqualification occurs as an indirect effect of having performed *put-p*. On the other hand, executing this action did not affect the fluent $\mathit{disq}(\mathit{put\text{-}p})$, which thus remains false according to the law of persistence. ■

3.2 ASSUMING QUALIFICATION BY DEFAULT

The intention of distinguishing a set of 'abnormality' fluents $\mathcal{F}_{ab}$ is to prefer among all suitable interpretations of domain descriptions those in which they are initially false. This would enable us to assume away abnormal circumstances whenever that is reasonable. Prior to discussing preference, however, we need to formalize the general notions of interpretation and model. Clearly, they both ought to respect the causal model Σ underlying the domain in question. Each interpretation (and model) contains a partial function *Res* which maps finite action sequences to states with the intended meaning that $\mathit{Res}([a_1, \ldots, a_n])$ would be the result of executing the action sequence $[a_1, \ldots, a_n]$ in the initial state (which itself is determined by $\mathit{Res}([])$).

[8] For the moment we neglect the possibility of miraculous disqualifications, which will be discussed later, in Section 3.3.

[9] Instead of explicitly providing the "only-if" part in (8), i.e., $\mathit{disq}(a) \supset \bigvee_{i \in I_a} f_i$, this could be implicitly obtained through circumscribing [McCarthy, 1980] the predicate *disq* in a given set of domain constraints; c.f. [Lifschitz, 1987], where this idea is applied to strict preconditions of actions.

Definition 7 Let Σ be the causal model determined by a domain description with domain constraints $\mathcal{D}$. A pair (Res, Σ) is an *interpretation* for this domain iff Res is a partial mapping from finite sequences of action names to states such that the following holds:

1. $Res([])$ is defined and satisfies $\mathcal{D}$.
2. For any finite sequence $[a_1, \ldots, a_{n-1}, a_n]$ of action names ($n > 0$), $Res([a_1, \ldots, a_{n-1}, a_n])$ is defined iff
   (a) $Res([a_1, \ldots, a_{n-1}])$ is defined;
   (b) $\overline{disq(a_n)} \in Res([a_1, \ldots, a_{n-1}])$; and
   (c) $\Sigma(a_n, Res([a_1, \ldots, a_{n-1}])) \neq \{\}$

   If it is defined, then $Res([a_1, \ldots, a_{n-1}, a_n])$ is a successor of $Res([a_1, \ldots, a_{n-1}]))$ and a_n. ■

If $Res([a_1, \ldots, a_n])$ is defined, we also say that the action sequence $[a_1, \ldots, a_n]$ is *qualified*. Then Definition 7 states that $[a_1, \ldots, a_{n-1}, a_n]$ is qualified if so is $[a_1, \ldots, a_{n-1}]$, if the state $Res([a_1, \ldots, a_{n-1}])$ does not imply an abnormal disqualification of a_n — which is indicated by fluent $disq(a_n)$ being false in this state—, and if all strict preconditions of a_n are met—which implies the existence of a successor state of a_n and $Res([a_1, \ldots, a_{n-1}])$. Notice that all defined function values of Res necessarily satisfy the underlying domain constraints if $Res([])$ does.

Based on the given a set of observations, an interpretation for a domain is considered a model iff all the observations hold in that interpretation.

Definition 8 Let Σ be the causal model of a domain description with observations $\mathcal{O}$. An interpretation (Res, Σ) is a *model of* $\mathcal{O}$ iff each observation in $\mathcal{O}$ holds in (Res, Σ), where

1. F **after** $[a_1, \ldots, a_n]$ is said to *hold* in (Res, Σ) iff $Res([a_1, \ldots, a_n])$ is defined and F is true in $Res([a_1, \ldots, a_n])$;
2. a **disqualified after** $[a_1, \ldots, a_n]$ is said to *hold* in (Res, Σ) iff $Res([a_1, \ldots, a_n])$ is defined but $Res([a_1, \ldots, a_n, a])$ is not. ■

Example 1 (continued) Let Σ be the causal model determined by the action laws (2), the domain constraints (1) and (9), and the causal relationships (4) and (10). Suppose given the observation

$$\overline{runs} \textbf{ after } [] \tag{12}$$

and consider, say, these two initial states:

$$\begin{aligned} Res_1([]) &= \{\overline{pot}, \overline{clog}, \overline{runs}, \overline{heavy}, \\ &\quad \overline{disq(start)}, \overline{disq(put\text{-}p)}\} \\ Res_2([]) &= \{\overline{pot}, \overline{clog}, \overline{runs}, heavy, \\ &\quad \overline{disq(start)}, disq(put\text{-}p)\} \end{aligned} \tag{13}$$

The corresponding interpretations[10] (Res_1, Σ) and (Res_2, Σ) satisfy (12), hence are models. Notice, however, that no 'abnormality' fluent is true in $Res_1([])$, as opposed to $Res_2([])$. Since $disq(start)$ holds in $\Sigma(put\text{-}p, Res_1([]))$ (c.f. (11)), the model (Res_1, Σ) entails that the engine cannot be ignited after putting a potato into the tail pipe. In contrast, the model (Res_2, Σ) is the formal counterpart of the counterintuitive conclusion where the action *put-p* is assumed to be abnormally disqualified in the first place. ■

While an interpretation must satisfy the given observations in order to constitute a model, this criterion alone does not suffice to assume away abnormal disqualifications. Obviously, the addition of observations can only decrease the set of models, never produce new ones. Consequently, if one defines an entailment relation stating that an observation is entailed by a set of observations if the former holds in all models of the latter, then this relation is monotone. Under the name *restricted monotonicity*, in [Lifschitz, 1993] this property is claimed generally desirable in theories of actions. Yet this is no longer appropriate when being confronted with the qualification problem because additional observations, such as detecting a potato in the tail pipe, may force us to withdraw previous (default) conclusions, like the conclusion that we are able to start the engine. We achieve this formally by a preference relation among the set of models, with the intention to select those which initially minimize truth of fluents in $\mathcal{F}_{ab}$ to the largest possible extent. When talking about entailment, attention is then restricted to models which are preferred in this sense. The following definition constitutes the core of our formal characterization of the qualification problem:

Definition 9 Let $\mathcal{F} \supseteq \mathcal{F}_{ab}$ be the underlying set of fluent names and $\mathcal{O}$ the set of observations of a domain description with causal model Σ. An interpretation $M' = (Res', \Sigma)$ is *less abnormal* than an interpretation $M = (Res, \Sigma)$, written $M' \prec M$, iff $Res'([]) \cap \mathcal{F}_{ab} \subsetneq Res([]) \cap \mathcal{F}_{ab}$.

A model M of $\mathcal{O}$ is *preferred* iff there is no model M' of $\mathcal{O}$ such that $M' \prec M$. An observation o is *entailed*, written $\mathcal{O} \mid\!\sim_\Sigma o$, iff o holds in each preferred model of $\mathcal{O}$. ■

In words, the less fluents in $\mathcal{F}_{ab}$ occur affirmatively in the initial state in a model the better. Obviously, the induced entailment relation, $\mid\!\sim_\Sigma$, is nonmonotonic as the addition of observations may change the set of preferred models entirely. In the sequel, we illustrate how this formal account of the qualification problem satisfies all the requirements which we demanded in the introduction.

[10] Notice that if all actions in a domain are deterministic (that is, each $\Sigma(a, S)$ is singleton or empty), then an interpretation (Res, Σ) is uniquely characterized by its initial state, $Res([])$.

3.2.1 How To Assume Away Disqualifications

The fundamental issue with the qualification problem is to assume away abnormal disqualifications by default. This, however, should concern only those disqualifications which do not admit a causal explanation. Our key example, in particular, is now treated in the expected way. Namely, any potential abnormal disqualification preventing us from putting a potato into the tail pipe is assumed away, for there is no evidence to the contrary. Likewise, any abnormal disqualification preventing us from starting the engine is assumed away as regards the initial state, whereas an abnormal disqualification of this very action after the insertion of a potato follows from the causal model without the necessity of granting abnormal circumstances.

Example 1 (continued) Recall from (13) the two models $M_1 = (Res_1, \Sigma)$ and $M_2 = (Res_2, \Sigma)$ of (12). Clearly, we have $M_1 \prec M_2$ due to $Res_1([]) \cap \mathcal{F}_{ab} = \{\}$ and $Res_2([]) \cap \mathcal{F}_{ab} = \{heavy, disq(put\text{-}p)\}$. Since each 'abnormality' fluent is false in the initial state in M_1, the latter obviously constitutes the unique preferred model. Whatever holds in M_1 is thus entailed by the domain. In particular, we have seen in (11) that $disq(start) \in Res_1([put\text{-}p])$. This implies that $[put\text{-}p, start]$ is not qualified in M_1, which in turn sanctions the entailment of

$$start \ \textbf{disqualified after} \ [put\text{-}p]$$

This constitutes the intended solution: The first action, *put-p*, is qualified by default and, as a consequence, action *start* is unqualified afterwards. ■

3.2.2 How To Explain Disqualifications

Aside from assuming away abnormal disqualifications of actions by default, one naturally seeks conceivable explanations in case a disqualifications has been—unexpectedly—observed without an apparent cause. Each preferred model that contains an abnormal disqualification also includes, provided the underlying domain constraints support this, a particular explanation. For otherwise the domain constraints would be violated in the state in which the disqualification occurs, as the following example illustrates.

Example 3 We extend the set of fluent names of Example 1 by *no-gas*, *low-batt*, and *engine-problem*, each of which shall belong to the subset $\mathcal{F}_{ab}$. These fluent names are combined in this domain constraint:

$$disq(start) \equiv clog \vee no\text{-}gas \vee low\text{-}batt \vee engine\text{-}problem$$

which shall replace the first formula in (9). Now suppose we are in a state where the engine is not running and where we also know that the tail pipe is not clogged nor is the tank empty, but nonetheless we encounter difficulties with starting the engine. The corresponding observations, i.e.,

$$\overline{runs} \ \textbf{after} \ []$$
$$\overline{clog} \wedge \overline{no\text{-}gas} \ \textbf{after} \ []$$
$$start \ \textbf{disqualified after} \ []$$

admit two preferred models: Each model (Res, Σ) must satisfy $disq(start) \in Res([])$ since $[start]$ is unqualified, according to the third observation, although the only strict precondition of *start*, viz. $\overline{runs}$, is initially true according to the first observation. Given $disq(start) \in Res([])$, the above domain constraint requires an additional 'abnormality' fluent be initially true in any model. The second observation excludes both *clog* and *no-gas*. Hence, a preferred model satisfies either $low\text{-}batt \in Res([])$ or else $engine\text{-}problem \in Res([])$. This in turn sanctions the entailment of the observation

$$low\text{-}batt \vee engine\text{-}problem \ \textbf{after} \ [] \qquad (14)$$

That is, problems with the battery or problems with the engine explain the observed abnormal disqualification of *start*. ■

3.2.3 How To Deal With Non-Determinism

The failure of the *chronological ignorance* approach to the qualification problem [Shoham, 1987; Shoham, 1988] in case of non-deterministic actions demonstrates a crucial difficulty with combining both abnormal disqualifications and non-determinism. The problem occurs whenever non-deterministic information provides sufficient evidence for an abnormal disqualification without, by virtue of being non-deterministic, necessitating it. Any formalism by which abnormal circumstances are negated whenever they do not provably hold, ignores uncertain evidence and, in so doing, supports unsound conclusions. As the Tail Pipe Marauder example will illustrate, our formal characterization of the qualification problem does not interfere with non-deterministic information and treats the latter in the appropriate, namely, the cautious way.

Example 2 (continued) Suppose given the observation

$$\overline{runs} \ \textbf{after} \ []$$

Since it is consistent with the observation to consider initially false all members of $\mathcal{F}_{ab}$, any preferred model (Res, Σ) must satisfy

$$Res([]) = \{\overline{pot}, \overline{clog}, \overline{runs}, \overline{disq(wait)}, \overline{disq(start)}\}$$

The action *wait* being non-deterministic (c.f. (3)), we know that either $Res([wait]) = Res([])$ or else $Res([wait]) = \{pot, clog, \overline{runs}, \overline{disq(wait)}, disq(start)\}$ holds in preferred models. Therefore, nothing definite follows about the status of the tail pipe, hence of the qualification of *start*, after performing $[wait]$. Consequently, the observation *runs* **after** $[wait, start]$, say, is not entailed, as intended. ■

3.3 MIRACULOUS DISQUALIFICATIONS

Thus far our theory supports generating explanations for surprising disqualifications by selecting among the conceivable reasons for this abnormality. Yet whenever the domain description renders invalid each of these explanations, then that goes beyond the capacity of the theory. Suppose given, as an example, the two observations

$$\begin{array}{l} \textit{start} \ \textbf{disqualified after} \ [] \\ \textit{runs} \ \textbf{after} \ [\textit{wait}, \textit{start}] \end{array} \qquad (15)$$

where *wait* is assumed to have no effects at all on the underlying fluents. No however (*a priori*) 'unlikely' model exists which simultaneously satisfies both of the observations. The reason is that any abnormality explaining the first disqualification necessarily transfers to the state after waiting, which contradicts the following success of performing *start*. Nonetheless, such situations, where the available explanations are insufficient to account for surprising disqualifications, are well conceivable and just prove our lack of omniscience.

We therefore need to extend our formalism to allow for observed yet inexplicable, in the above sense, action disqualifications. To this end, the formal notions of interpretation and model are enhanced by a component accommodating these so-called *miraculous* disqualifications. As we have seen, a miraculous disqualification may appear or disappear even though the truth values of the fluents suggest identical states. This is why any such disqualification is to be associated with the sequence of actions after whose execution it occurs, rather than with the respective state. Formally, the new component, denoted by Υ, consists of non-empty action sequences indicating the following: Whenever $[a_1, \ldots, a_{n-1}, a_n] \in \Upsilon$ $(n > 0)$, then action a_n is disqualified in the state resulting from performing $[a_1, \ldots, a_n]$ even if all strict preconditions of a_n and also $\overline{\textit{disq}(a_n)}$ hold in that state. The following extends Definition 7 accordingly.

Definition 10 Let Σ be the causal model determined by a domain description with domain constraints $\mathcal{D}$. A triple $(\textit{Res}, \Sigma, \Upsilon)$ is an *interpretation* for this domain iff Υ is a set of non-empty, finite sequences of action names and *Res* is a partial mapping from finite sequences of action names to states such that the following holds:

1. $\textit{Res}([])$ is defined and satisfies $\mathcal{D}$.
2. For any finite sequence $[a_1, \ldots, a_{n-1}, a_n]$ of action names $(n > 0)$, $\textit{Res}([a_1, \ldots, a_{n-1}, a_n])$ is defined iff
   (a) $\textit{Res}([a_1, \ldots, a_{n-1}])$ is defined;
   (b) $\overline{\textit{disq}(a_n)} \in \textit{Res}([a_1, \ldots, a_{n-1}])$;
   (c) $\Sigma(a_n, \textit{Res}([a_1, \ldots, a_{n-1}])) \neq \{\}$; and
   (d) $[a_1, \ldots, a_{n-1}, a_n] \notin \Upsilon$.

   If it is defined, then $\textit{Res}([a_1, \ldots, a_{n-1}, a_n])$ is a successor of $\textit{Res}([a_1, \ldots, a_{n-1}]))$ and a_n. ■

The additional clause, 2(d), states that a sequence of actions can only be qualified if it is not miraculously disqualified. As before, a model of a set of observations is an interpretation in which all the observations hold (c.f. Definition 8).

Example 4 The domain discussed in Example 1 is extended by the action name *wait* in conjunction with the action law $\langle\{\}, \textit{wait}, \{\}\rangle$. Furthermore, suppose given the aforementioned observations (15). While no model $(\textit{Res}, \Sigma, \Upsilon)$ with $\Upsilon = \{\}$ exists for this domain, as argued above, both these observations hold in the interpretation $(\textit{Res}, \Sigma, \Upsilon)$ where $\textit{Res}([])$ is

$$\{\overline{\textit{pot}}, \overline{\textit{clog}}, \overline{\textit{runs}}, \overline{\textit{heavy}}, \overline{\textit{disq}(\textit{start})}, \overline{\textit{disq}(\textit{put-p})}\} \qquad (16)$$

and $\Upsilon = \{[\textit{start}]\}$. This interpretation thus constitutes a model. ■

Clearly, miraculous disqualifications, too, are to be minimized to the largest possible extent. Moreover, miraculous disqualifications are meant as means to account for abnormal disqualifications which do not admit an explanation even by granting abnormal circumstances. As such, miraculous disqualifications need to be minimized with higher priority. As opposed to explicable disqualifications, miraculous ones can well be minimized globally, that is, without worrying about causality—would they admit a causal explanation they would not be miraculous. We thus arrive at the following extension of our preference criterion:

Definition 11 Let $\mathcal{F} \supseteq \mathcal{F}_{ab}$ be the underlying set of fluent names and $\mathcal{O}$ the set of observations of a domain description with causal model Σ. An interpretation $M' = (\textit{Res}', \Sigma, \Upsilon')$ is *less abnormal* than an interpretation $M = (\textit{Res}, \Sigma, \Upsilon)$, written $M' \prec M$, iff

1. either $\Upsilon' \subsetneqq \Upsilon$,
2. or $\Upsilon' = \Upsilon$ and $\textit{Res}'([]) \cap \mathcal{F}_{ab} \subsetneqq \textit{Res}([]) \cap \mathcal{F}_{ab}$.

The notions of preferred model and entailment in Definition 9 modify accordingly. ■

Example 4 (continued) We have seen that the domain considered above does not admit a model without miraculous disqualifications. It follows that the above model $M = (\textit{Res}, \Sigma, \Upsilon)$—where $\textit{Res}([])$ is as in (16) and $\Upsilon = \{[\textit{start}]\}$—is preferred, for it declares a single action sequence miraculously disqualified and negates each 'abnormality' fluent in the initial state. As a matter of fact, M is the only preferred model since any model $(\textit{Res}', \Sigma, \Upsilon')$ must satisfy $[\textit{start}] \in \Upsilon'$ and also $\overline{\textit{runs}} \in \textit{Res}'([])$ (the latter is due to $[\textit{wait}, \textit{start}]$ being qualified according to (15)). ■

4 FLUENT CALCULUS AND THE QUALIFICATION PROBLEM

Finally, we briefly sketch a suitable action calculus which is capable of handling abnormal action disqualifications. Our encoding employs the representation technique underlying the *fluent calculus* [Hölldobler and Schneeberger, 1990; Thielscher, 1997]. As opposed to the situation calculus [McCarthy and Hayes, 1969; Reiter, 1991], the fluent calculus employs structured state terms, each of which consists in a collection of all fluent literals that are true in the state being represented. To this end, fluent literals are reified, i.e., formally represented as terms. An example state term is $\mathit{in\text{-}pipe}(\mathit{potato}) \circ \overline{\mathit{heavy}(\mathit{potato})} \circ \mathit{clog}$ [11] where the bar denoting negative fluent expressions is formally a unary function and where $\circ$ denotes a special binary function symbol which obeys the laws of associativity and commutativity. It has first been argued in [Hölldobler and Schneeberger, 1990] that this representation technique avoids extra axioms (e.g., frame axioms [McCarthy and Hayes, 1969]) to encode the general law of persistence: The effects of actions are modeled by manipulating state terms through removal and addition of sub-terms. Then all sub-terms which are not affected by these operations remain in the state term, hence continue to be true.

Our solution to the qualification problem in the fluent calculus builds on the integration of causal relationships into this calculus [Thielscher, 1997]. While the fluent calculus provides monotonic solutions to both the frame problem as well as the ramification problem, the qualification problem, as we have seen, necessitates some kind of nonmonotonicity. In particular, we employ for each 'abnormality' fluent name $f_{ab} \in \mathcal{F}_{ab}$ the *default rule* [Reiter, 1980]

$$\frac{: \forall s\,[\mathit{Initial}(s) \supset \neg \mathit{Holds}(f_{ab}(x_1, \ldots, x_n), s)]}{\forall s\,[\mathit{Initial}(s) \supset \neg \mathit{Holds}(f_{ab}(x_1, \ldots, x_n), s)]}$$

This rule should be read as: Provided it is consistent, conclude that if s represents the initial state then an instance $f_{ab}(t_1, \ldots, t_n)$ is false in s. In addition, miraculous disqualifications are assumed away, whenever possible, by applying defaults of the form

$$\frac{: \neg \mathit{Miracle}(a^*)}{\neg \mathit{Miracle}(a^*)} \quad (a^* \text{ action sequence})$$

Since miraculous disqualifications are to be minimized with higher priority, we employ the concepts of *Prioritized Default Logic* [Brewka, 1994]. The report [Thielscher, 1996] contains full details as well as a formal proof of the adequacy of this extension with regard to the theory developed in Section 3.

[11] As opposed to the formal language used in the preceding sections, our action calculus supports non-propositional fluents, such as *in-pipe*, whose arguments are chosen from a set of *entities*, such as *potato*.

5 DISCUSSION

We have proposed a formal characterization of the qualification problem from the perspective that requiring global minimization of abnormal disqualifications is obviously inadequate. Our theory may be summarized as follows. Any domain description is supposed to contain a distinguished set of fluents $\mathcal{F}_{ab}$, each of which describes abnormal circumstances and thus is to be assumed false by default. This assumption, however, needs to be restricted to the initial state, so that these fluents are subject to the general law of persistence but are also potentially (directly or indirectly) affected by the execution of actions. Among these 'abnormality' fluents are propositions, denoted $\mathit{disq}(a)$, which state that an action a is abnormally disqualified. Domain constraints relating these fluents with possible causes of an abnormal disqualification support the proliferation of explanations in case an abnormal disqualification—surprisingly—occurs. In addition, miraculous disqualifications accommodate situations in which a suitable explanation cannot be provided. The default assumption of 'normality' is formally represented by a model preference criterion (Definition 11), which induces a nonmonotonic entailment relation among observations.

Using a suitably simple action language, the focus in this paper has been on the qualification problem. The underlying principles of our theory, however, are sufficiently fundamental and general to not depend on this specific language. Thus these principles could equally well be employed in other, more elaborated formal theories of actions like, e.g., [Gelfond and Lifschitz, 1993; Sandewall, 1994; Thielscher, 1995], in view of the qualification problem. Likewise, existing action calculi may be enhanced on this basis in order that they become capable of dealing with abnormal action disqualifications. As an example, we have sketched a way to embed the fluent calculus in an appropriate nonmonotonic theory. The adequacy of the resulting framework has been established by relating it to our formal characterization of the qualification problem. This adds another item to the list of ontological aspects which the fluent calculus is capable of dealing with, such as non-deterministic and concurrent actions [Bornscheuer and Thielscher, 1997], indirect effects of actions [Thielscher, 1997], and continuous change [Hermann and Thielscher, 1996].

Besides the proposal pursued in this paper, the only existing alternative to global minimization of abnormalities as a solution to the qualification problem is the concept of *chronological ignorance* [Shoham, 1987; Shoham, 1988]. Roughly speaking, the crucial idea there is to assume away, by default, abnormal circumstances which do not provably hold, and simultaneously to prefer minimization of abnormalities at earlier timepoints. This approach treats our introductory key example correctly. The interesting, albeit informal,

reason for coming to the desired conclusion in this and similar cases is a certain respect of causality hidden in this method: By minimizing chronologically, one tends to minimize causes rather than effects—which is the right thing to do—simply because in general causes precede effects. On the other hand, it has already been shown elsewhere (e.g., [Kautz, 1986; Sandewall, 1993; Stein and Morgenstern, 1994]) that the applicability of chronological minimization is intrinsically restricted to domains which do not include non-deterministic information. The Tail Pipe Marauder scenario of Example 2 constitutes a simple domain which does not fall into that category. Given that non-deterministically there might or might not be a potato in the tail pipe, chronological ignorance sanctions the prediction that nonetheless starting the engine will be successful. For it cannot be *proved* that this action has an abnormal disqualification—which thus is assumed away. While the qualification problem means to assume away abnormal circumstances whenever they do not provably hold, the Tail Pipe Marauder domain illustrates that this approach is in general too optimistic if the execution of a non-deterministic action renders quite possible such circumstances. In contrast, our characterization of the qualification problem accounts for this as the minimization procedure applied to abnormal or miraculous disqualifications does not interfere with the results of non-deterministic actions.

Our approach to the qualification problem shares with *Motivated Action Theory* [Stein and Morgenstern, 1994] the insight that an appropriate notion of causality is necessary when assuming away abnormalities. In the latter framework, occurrences of actions and events are assumed away by default while considering the possibility that they are caused (or, in other words, *motivated*, hence the name). This minimizing unmotivated events and our minimizing non-caused abnormal disqualifications are somehow complementary while based on similar principles. Of course, the formal realizations are quite different. An unsatisfactory property of Motivated Action Theory is that the preference criterion, that is, *motivation*, depends on the syntactical structure of the formulas representing causal knowledge. As a consequence, logical equivalent formalizations may induce different preference criteria, of which only one is the desired. Moreover, the formal concept of motivation becomes rather complicated in case of disjunctive (i.e., non-deterministic) information, which entails difficulties with assessing its range of applicability.

Throughout the paper, we have taken action disqualifications as rendering physically impossible the execution of the respective action. A desirable refinement is to consider actions be disqualified *as to producing a certain effect* (c.f. [Gelfond *et al.*, 1991], e.g.). This is accomplished with a simple, straightforward extension of our theory. In addition to the fluents $\mathit{disq}(a)$, we introduce fluents of the form $\mathit{disq}(a, \ell)$, whose intended reading is "action a fails to produce effect ℓ." These fluents, too, belong to the set $\mathcal{F}_{ab}$ and may be related to other 'abnormality' fluents by means of domain constraints, like in

$$\mathit{disq}(\mathit{shoot}, \overline{\mathit{alive}}) \equiv \mathit{bad\text{-}sight} \vee \mathit{bad\text{-}shooter} \vee \mathit{bad\text{-}gun}$$

Suppose, then, $\langle C, a, E\rangle$ is the action law to be applied to some state S. The effect which a actually manages to produce if performed in S is formally given by $E' := E \setminus \{\ell : \mathit{disq}(a, \ell) \in S\}$. Let $C' := C \setminus \{\ell, \overline{\ell} : \ell \in E \setminus E'\}$, which guarantees that $|C'| = |E'|$, then $(S \setminus C') \cup E'$ is taken as the intermediate state which is subject to the following ramification process. The notion of a successor state modifies accordingly while all further concepts, viz. interpretations, models, and the preference criterion, remain unaltered.

Finally, it needs to be mentioned that we gave emphasis only to the representational aspect of the qualification problem, as opposed to the computational aspect. That the latter is of equal importance has been pointed out, e.g., in [Elkan, 1995]. Our analysis has revealed some hitherto unnoticed problems with the representational aspect and, to state the obvious, the computational aspect cannot be pursued without an appropriate representation of the problem. Named the computational part of the qualification problem, the challenge is to find a computational model that enables the reasoning agent to assume that an action be qualified without even *thinking* of all possible disqualifying causes—unless some piece of knowledge hints at their presence. In principle, the special fluents $\mathit{disq}(a)$ employed in our theory serve this purpose: By assuming $\overline{\mathit{disq}(a)}$, one jumps to the conclusion that a be qualified provided all strict preconditions are met. Still, on the other hand, in order that this assumption be justified, its consistency as regards the underlying domain constraints must be guaranteed. In a standard reasoning system, this in turn involves consideration (and exclusion) of all the potential disqualifying, abnormal circumstances. A solution to the computational part of the qualification problem thus requires a different computational model, presumably based on some parallel architecture, by which all related domain constraints are ignored unless they are explicitly 'activated' by some piece of information. Although this aspect was not among the topics of this paper, the foundations have been laid.

Acknowledgments

The author is grateful to Wolfgang Bibel, Christoph Herrmann, and two anonymous referees for helpful comments and suggestions, and to Erik Sandewall for enlightening remarks on the subject of the paper.

References

[Bornscheuer and Thielscher, 1997]
S. Bornscheuer and M. Thielscher. Explicit and implicit indeterminism: Reasoning about uncertain and contradictory specifications of dynamic systems. *J. of Logic Programming*, 1997.

[Brewka, 1994] G. Brewka. Adding priorities and specificity to default logic. In C. MacNish, D. Pearce, and L. Pereira, eds., *Proc. of the European Workshop on Logics in AI (JELIA)*, vol. 838 of *LNAI*, p. 50–65. Springer 1994.

[Elkan, 1995] C. Elkan. On solving the qualification problem. In C. Boutilier and M. Goldszmidt, eds., *Extending Theories of Actions: Formal Theory and Practical Applications*, vol. SS–95–07 of *AAAI Spring Symposia*, Stanford University, 1995.

[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.

[Gelfond and Lifschitz, 1993] M. Gelfond and V. Lifschitz. Representing action and change by logic programs. *J. of Logic Programming*, 17:301–321, 1993.

[Gelfond *et al.*, 1991] M. Gelfond, V. Lifschitz, and A. Rabinov. What are the limitations of the situation calculus? In S. Boyer, ed., *Automated Reasoning, Essays in Honor of Woody Bledsoe*, p. 167–181. Kluwer Academic, 1991.

[Ginsberg and Smith, 1988a] M. Ginsberg and D. Smith. Reasoning about action I: A possible worlds approach. *Artificial Intelligence*, 35:165–195, 1988.

[Ginsberg and Smith, 1988b] M. Ginsberg and D. Smith. Reasoning about action II: The qualification problem. *Artificial Intelligence*, 35:311–342, 1988.

[Hanks and McDermott, 1987] S. Hanks and D. McDermott. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33(3):379–412, 1987.

[Herrmann and Thielscher, 1996] C. Herrmann and M. Thielscher. Reasoning about continuous processes. In B. Clancey and D. Weld, eds., *Proc. of the AAAI*, p. 639–644, Portland, 1996.

[Hölldobler and Schneeberger, 1990]
S. Hölldobler and J. Schneeberger. A new deductive approach to planning. *New Generation Computing*, 8:225–244, 1990.

[Kautz, 1986] H. Kautz. The logic of persistence. In *Proc. of the AAAI*, p. 401–405, Philadelphia, 1986.

[Lifschitz, 1987] V. Lifschitz. Formal theories of action (preliminary report). In J. McDermott, ed., *Proc. of the IJCAI*, p. 966–972, Milan, 1987.

[Lifschitz, 1993] V. Lifschitz. Restricted monotonicity. In *Proc. of the AAAI*, p. 432–437, Washington, 1993.

[Lin and Reiter, 1994] F. Lin and R. Reiter. State constraints revisited. *J. of Logic and Computation*, 4(5):655–678, 1994.

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

[McCarthy, 1977] J. McCarthy. Epistemological problems of artificial intelligence. In *Proc. of the IJCAI*, p. 1038–1044, Cambridge, 1977.

[McCarthy, 1980] J. McCarthy. Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[McCarthy, 1986] J. McCarthy. Applications of circumscription to formalizing common-sense knowledge. *Artificial Intelligence*, 28:89–116, 1986.

[Reiter, 1980] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[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, ed., *Artificial Intelligence and Mathematical Theory of Computation*, p. 359–380. Academic Press, 1991.

[Sandewall, 1993] E. Sandewall. Systematic assessment of temporal reasoning methods for use in autonomous systems. In B. Fronhöfer, ed., *Workshop on Reasoning about Action & Change at IJCAI*, p. 21–36, Chambéry, 1993.

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

[Shoham, 1987] Y. Shoham. *Reasoning about Change.* MIT Press, 1987.

[Shoham, 1988] Y. Shoham. Chronological ignorance: Experiments in nonmonotonic temporal reasoning. *Artificial Intelligence*, 36:279–331, 1988.

[Stein and Morgenstern, 1994] L. 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. Mellish, ed., *Proc. of the IJCAI*, p. 1956–1962, Montreal, 1995.

[Thielscher, 1996] M. Thielscher. Qualification and Causality. Technical Report TR-96-026, ICSI, Berkeley, July 1996. (Available at `http://kirmes.inferenzsysteme.informatik.th-darmstadt.de/~mit`; click the item `Technical Reports`).

[Thielscher, 1997] M. Thielscher. Ramification and causality. *Artificial Intelligence*, 1997. (To appear. A preliminary version is available as Technical Report TR-96-003, ICSI, Berkeley, Jan. 1996).

Reasoning about Discontinuities in the Event Calculus

Rob Miller
Department of Computing,
Imperial College of Science, Technology & Medicine,
180 Queen's Gate, London SW7 2BZ, U.K.
rsm@doc.ic.ac.uk
http://www.doc.ic.ac.uk/~rsm/

Murray Shanahan
Department of Computer Science,
Queen Mary & Westfield College,
Mile End Road, London E1 4NS, U.K.
mps@dcs.qmw.ac.uk
http://www.dcs.qmw.ac.uk/~mps/

Abstract

This paper describes a logic-based formalism which combines techniques for reasoning about actions with standard mathematical techniques for modelling dynamic systems using the differential calculus. The formalism inherits a robust solution to the frame problem which can handle concurrency, non-determinism, domain constraints and narrative. It also incorporates a mechanism for reasoning about the boundary conditions associated with systems of differential equations defined over various intervals. This mechanism overcomes a number of drawbacks of previous systems.

1 INTRODUCTION

Solutions to the frame problem now exist which can handle a variety of phenomena, such as narrative, concurrent action and non-deterministic action (see (Shanahan 1997) for a general discussion). However, the topic of continuous change has received relatively little attention in the Reasoning about Action literature. In particular, a satisfactory general framework has yet to be developed which reconciles logic-based techniques for reasoning about action with the standard mathematical approach to modelling dynamic systems, using differential calculus.

Some previous logic-based approaches to reasoning about action do allow limited types of mathematical expressions involving continuously varying parameters to be embedded within domain descriptions. For example, Shanahan (1990) presents a logic programming approach to representing continuous change, based on the event calculus of Kowalski and Sergot (1986). Further work along these lines has been done by Van Belleghem *et al.* (1994), whose treatment allows a wider class of mathematical expressions, by Shanahan (1995), where a full predicate calculus version using circumscription is presented, and by Herrmann and Thielscher (1996), whose notion of a *process* generalises Shanahan's notion of a *trajectory*. However, none of these frameworks incorporates the notion of a derivative function. Hence any differential equations representing the domain have to be solved before they can be added to the model.

In contrast, Sandewall (1989a, 1989b) presents an approach to continuous change which combines logic and differential equations. Sandewall identifies the need to incorporate mechanisms within such frameworks to deal with the boundary conditions usually associated with sets of such equations. He advocates a default reasoning method, based on Shoham's notion of chronological minimisation (Shoham 1988), to generate new boundary conditions when an action or event transforms one mathematical model into another. However, as shown by Rayner (1991), Sandewall's approach leaves open the possibility of anomalous models. Miller (1996) introduces techniques for incorporating differential equations within a circumscribed situation calculus. He avoids Rayner's anomalous models through the use of a (minimised) *Breaks* predicate, which describes when actions cause discontinuities in particular parameters.

This paper draws on the techniques of (Miller 1996) in order to construct an event calculus resembling that of (Shanahan 1997, Ch. 16), but which allows continuous change to be described using arbitrary systems of differential equations. Unlike existing versions of the situation calculus which can handle continuous change, such as (Miller 1996) and (Reiter 1996), this calculus inherits the ability to handle domain constraints, concurrent actions, and actions with non-deterministic effects from the event calculus (Shanahan 1997, Ch. 16).

2 TWO EXAMPLES

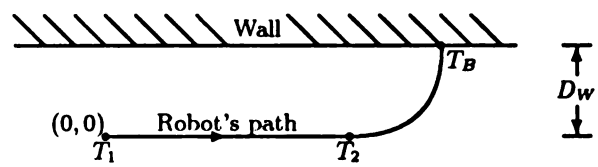
Both the examples below help to illustrate how arbitrary systems of simultaneous differential equations can be incorporated within an axiomatic description of a given domain. The first, adapted from (Shanahan 1996), concerns the movement of a mobile robot, which will stop whenever it bumps against an object (in this case, a wall). It thus shows how the formalism represents actions (events) which are "triggered" when certain conditions occur. The second is a modification of the "water tanks" example from (Miller 1996). It shows how the formalism supports reasoning about concurrently performed actions (a tap is opened, and some water is simultaneously scooped from a tank). A key feature of both these examples is that, as is usual in the mathematical modelling of dynamic systems, the continuously changing aspects of the domain are represented as a system of (possibly *simultaneous* or *coupled*) differential equations, together with one or more sets of *boundary* conditions (or *initial* conditions). In such models, the complete set of boundary conditions may not be listed – some conditions are implicit in the accompanying physical description of the domain, or are "common sense".

In this section we concentrate on showing how both examples can be modelled "conventionally" using differential calculus, highlighting some of the reasoning processes involved in obtaining explicit mathematical expressions for parameters' behaviours over time. In both cases we go into some mathematical detail, primarily in order to analyse why and how specific values are assigned to arbitrary constants of integration during the mathematical modelling process. This analysis is central to the understanding of the rest of the paper. As we shall see, these value assignments are based partly on "common sense", and it was incomplete modelling of this implicit assignment process which lead to problems with anomalous models in the work of Sandewall (1989a, 1989b).

In both examples, dashes (′) are used to refer to derivatives of parameters – for example, P' refers to the first derivative with respect to time of P.

2.1 THE MOBILE ROBOT EXAMPLE

The robot in this example can start (and stop) moving at any constant speed in the direction it is facing (its "bearing"). It can also (continuously) vary its bearing by turning at any constant angular velocity, possibly while moving. It can therefore follow any path consisting of a series of circle fragments and straight lines. It has sensors which will cause it to stop if it bumps against any object. We will represent the following scenario. At time 0 the robot is stationary at the point $(0,0)$, facing east, and has angular velocity 0. At time T_1 it starts moving forward (east) at speed S^+. At time T_2 it also starts to turn in a anticlockwise direction with angular speed A^+. Some time later it bumps into the only other object in the domain – a wall (of "infinite length") running east-west, at a distance D_W due north from the origin.



By idealising the robot as a moving point and the wall as a fixed line, a mathematical model of this domain can be formulated in a simple way, using the parameters (real-valued functions of time) *Bearing*, *NSP* ("North-South-Position"), and *EWP* ("East-West-Position"). We can write down the following mathematical constraints between these parameters using the "constants" S (forward speed) and A (angular speed), whose values are determined directly and instantaneously by the robot's actions, or by the event of bumping into the wall.

$$EWP'(t) = S.\cos(Bearing(t)) \quad (1)$$
$$NSP'(t) = S.\sin(Bearing(t)) \quad (2)$$
$$Bearing'(t) = A \quad (3)$$

Let T_B be the time that the robot bumps against the wall. We know that $EWP(0) = NSP(0) = Bearing(0) = 0$, that $NSP(T_B) = D_W$, and that, because of the robot's actions and the effect of bumping into the wall,

$$\text{for } 0 \leq t \leq T_1: \quad A = 0 \text{ and } S = 0 \quad (4)$$
$$\text{for } T_1 < t \leq T_2: \quad A = 0 \text{ and } S = S^+ \quad (5)$$
$$\text{for } T_2 < t \leq T_B: \quad A = A^+ \text{ and } S = S^+ \quad (6)$$
$$\text{for } T_B < t: \quad A = 0 \text{ and } S = 0 \quad (7)$$

The formalism described in this paper allows us to include equations (1)–(3) directly in the domain-dependent axiomatisation, rather than first having to solve them to determine an explicit or stratified set of expressions for *EWP*, *NSP* and *Bearing*, as in (Shanahan 1990) and related approaches. It also allows us to infer (4)–(7) from more general knowledge, both about the effects and timings of the robot's actions, and about the general hypothetical circumstances un-

der which the robot will bump into the wall.

It is important to note, however, that from a strictly mathematical point of view, expressions (1)–(7) do not contain enough information to uniquely determine the position of the robot at a given time T (even when considered together with information about the values of each parameter at time 0). An assumption also has to be made that *EWP*, *NSP* and *Bearing* are continuous at times T_1, T_2 and T_B. The simultaneous solution to (1)–(3) for any given value of S and A (obtained by integrating (3), substituting the answer in (1) and (2), and integrating again) is

$$EWP(t) = \frac{S}{A}\sin(At + C_1) + C_2$$
$$NSP(t) = \frac{-S}{A}\cos(At + C_1) + C_3$$
$$Bearing(t) = At + C_1$$

where C_1, C_2 and C_3 are arbitrary constants of integration. In the time interval $[0, T_1]$ (when $S = A = 0$), the values of these constants can be computed using knowledge about the initial values of the three parameters, but in the intervals $(T_1, T_2]$, $(T_2, T_B]$ and (T_B, ∞), C_1, C_2 and C_3 can only be given specific numerical values by making continuity assumptions. Fortunately, it is "common sense" that the robot cannot instantaneously shift its own position or bearing. However, it is not difficult to imagine another scenario where some external action (such as giving the robot a good shove from behind) *does* cause an instantaneous shift in position (at least at the level of detail at which we wish to model the domain).

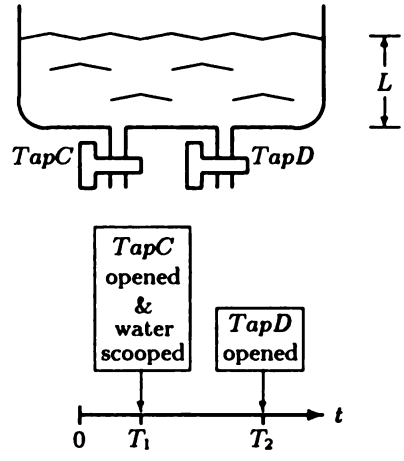
The formalism described below provides a general purpose default reasoning mechanism which allows us to infer the extra common sense information that the parameters *EWP*, *NSP* and *Bearing* are continuous at times T_1, T_2 and T_B (when particular actions or events cause discontinuities in some or all of their derivatives, or in other unrelated parameters), without forbidding, in principle, the existence of other actions or events that could potentially cause discontinuities in the robot's position or orientation. Putting all this together, it allows us to infer the explicit values of *EWP*, *NSP* and *Bearing* at any time t, and (assuming that $D_W < \frac{2S^+}{A^+}$, so that the robot doesn't turn in a full circle and miss the wall) it allows us to infer that

$$T_B = T_2 + \frac{1}{A^+}\arccos(1 - \frac{A^+ . D_W}{S^+})$$

The formalism is "elaboration tolerant" in the sense that, at some later date, we can represent new knowledge about other effects of the robot's actions, or about other occurrences of actions or events at specific times, simply by adding to the existing set of domain-dependent axioms.

2.2 THE WATER TANK EXAMPLE

This example involves an open-top water tank. In the bottom of the tank are two taps, *TapC* and *TapD*, both of which (when open) discharge water out of the tank at a rate proportional to the level of water in it (i.e. at a rate proportional to the water pressure at the bottom of the tank). Initially, both taps are closed, and the level of water in the tank is L. At time T_1, half the water is (instantaneously) scooped out of the tank, and, simultaneously, *TapC* is opened. At time T_2, *TapD* is also opened.



A mathematical model of this domain can be formulated using three parameters. $Level(t)$ is the function representing the level of water in the tank, and $FlowC(t)$ and $FlowD(t)$ represent the water flow through *TapC* and *TapD* respectively. Let K be the constant of proportionality between the level of water and the flow through either of the taps when open. The following equations are each applicable in the circumstances indicated

Always:	$Level'(t) = -(FlowC(t) + FlowD(t))$	(8)
TapC closed:	$FlowC(t) = 0$	(9)
TapC open:	$FlowC(t) = K.Level(t)$	(10)
TapD closed:	$FlowD(t) = 0$	(11)
TapD open:	$FlowD(t) = K.Level(t)$	(12)

In the interval $[0, T_1]$ the relevant equations are (8), (9) and (11). Their simultaneous solution is

$$Level(t) = C_4 \quad \text{(S1)}$$
$$FlowC(t) = 0 \quad \text{(S2)}$$
$$FlowD(t) = 0 \quad \text{(S3)}$$

where C_4 is an arbitrary constant of integration. In fact, since we have also been given the initial condition $Level(0) = L$, we know that $C_4 = L$.

At time T_1, $TapC$ is opened, so that in the interval $(T_1, T_2]$ the equations to solve are (8), (10) and (11). Their simultaneous solution is

$$Level(t) = C_5.e^{-Kt}$$
$$FlowC(t) = C_5.Ke^{-Kt}$$
$$FlowD(t) = 0$$

where C_5 is an arbitrary constant of integration. We also know that the water level in the tank is instantaneously decreased from L to $\frac{L}{2}$ at T_1 by a "scoop" action. Hence, with a little algebra, it is easy to see that $C_5 = \frac{L}{2}e^{KT_1}$. So the overall solution for $t \in (T_1, T_2]$ is

$$Level(t) = \tfrac{L}{2}e^{-K(t-T_1)} \quad \text{(S4)}$$
$$FlowC(t) = \tfrac{LK}{2}e^{-K(t-T_1)} \quad \text{(S5)}$$
$$FlowD(t) = 0 \quad \text{(S6)}$$

In the interval (T_2, ∞), both $TapC$ and $TapD$ are open, and the equations to solve are (8), (10) and (12). Their simultaneous solution is

$$Level(t) = C_6.e^{-2Kt}$$
$$FlowC(t) = C_6.Ke^{-2Kt}$$
$$FlowD(t) = C_6.Ke^{-2Kt}$$

where again C_6 is an arbitrary real valued constant.

What value should C_6 take? Everyday knowledge about taps and tanks tells us that $Level(t)$ is continuous at T_2. In other words, we know that by itself the action of turning on $TapD$ will only cause water to disappear from the tank gradually, not instantaneously. (Although, at the level of physical detail we have chosen to incorporate in our mathematical model, the action of scooping water from the tank *does* cause the water level to drop instantaneously.) Hence $C_6 = \frac{L}{2}e^{K(T_1+T_2)}$, and the overall solution for $t \in (T_2, \infty)$ is

$$Level(t) = \tfrac{L}{2}e^{K(T_1+T_2-2t)} \quad \text{(S7)}$$
$$FlowC(t) = \tfrac{LK}{2}e^{K(T_1+T_2-2t)} \quad \text{(S8)}$$
$$FlowD(t) = \tfrac{LK}{2}e^{K(T_1+T_2-2t)} \quad \text{(S9)}$$

Notice that, if instead we had wished to keep $FlowD$ and its derivatives continuous at T_2, we could have made the assignment $C_6 = 0$. Alternatively, if we had wished to keep the first derivatives $Level'$ and $FlowC'$ continuous, we could have made the assignment $C_6 = \frac{L}{4}e^{K(T_1+T_2)}$. The important general point here is that in many mathematical models, it is possible to trade discontinuities in some parameters for discontinuities in others, using alternative assignments of values to (arbitrary) constants of integration. It is only common sense, or extra knowledge about the physical reality of the domain being modelled, that allows us to pick the right assignments from the different sets of possibilities.

The formalism described below allows us to include equations (8)–(12), along with their conditions of applicability, directly in the domain-dependent axiomatisation. The default reasoning method it incorporates allows us to infer that there is a discontinuity in $Level$ at T_1 (an instantaneous change from the value L to the value $\frac{L}{2}$ caused by the action of scooping water from the tank), but that $Level$ is continuous at T_2 (although its derivatives are not). In doing so, it correctly eliminates the anomalous models of the domain (which would otherwise be sanctioned by (8)–(12)) in which all (or half) the remaining water in the tank instantaneously disappears at T_2, thus illegally "trading" the discontinuity in $FlowD$ (or the discontinuities in $Level'$ and $FlowC'$) at T_2 for discontinuities in $Level$ and $FlowC$. Hence it allows us to correctly infer the values of $Level$, $FlowC$, $FlowD$ and their derivatives at any time t.

3 AN EXTENDED EVENT CALCULUS

The event calculus presented in this section is written in a sorted predicate calculus, with sorts as summarised in the following table.

NAME OF SORT	SYMBOL	VARIABLES
Actions	$\mathcal{A}$	$a, a_1, a_2, \ldots$
Fluents	$\mathcal{F}$	$f, f_1, f_2, \ldots$
Times	$\mathcal{T}$	$t, t_1, t_2, \ldots$
Parameters	$\mathcal{P}$	$p, p_1, p_2, \ldots$
Reals	$\mathcal{R}$	$r, r_1, r_2, \ldots$
Domain objects	$\mathcal{X}$	$x, x_1, x_2, \ldots$

Models will be considered only in which terms of sort $\mathcal{R}$ and $\mathcal{T}$ are interpreted as real and non-negative real numbers respectively. The sorting of the predicate symbols[1] in the language can be understood from their

[1] As mentioned previously, this formalism builds on several previous axiomatisations. However, some predicate and function names have been changed for the sake of clarity. *InitialisedTrue* and *InitialisedFalse* correspond to $Initially_P$ and $Initially_N$ in (Shanahan 1997), and *BreaksTo* and *Value* are analogous to *InstantEffect* and *Function* in (Miller 1996).

arguments in the axioms below, in which all variables are assumed to be universally quantified with maximum scope unless otherwise stated. The function symbols $Value : \mathcal{P} \times \mathcal{T} \mapsto \mathcal{R}$ and $\delta : \mathcal{P} \mapsto \mathcal{P}$ are also introduced. The term $Value(\delta(P), T)$ represents the numerical value of the first derivative of parameter P at time T.

The six core event calculus axioms (EC1)–(EC6) below, which do not directly concern continuous change, are domain-independent, i.e. included in every theory. For the sake of generality, this part of the axiomatisation includes a mechanism, inspired by Kartha and Lifschitz's work (1994) and Sandewall's notion of *occlusion* (Sandewall 1994), for dynamically adding fluents to (and removing fluents from) the "frame" (i.e. the set of fluents subject to the "commonsense law of inertia"). It was first introduced into the event calculus by Shanahan (1995). At time 0, exactly those fluents which have been *initialised true* or *initialised false* belong to the frame. These mechanisms are useful, for example, in dealing with domain constraints and non-deterministic actions (Shanahan 1995). However, in the example domains used in this paper, all fluents are permanently subject to the commonsense law of inertia, so that the (universally quantified) sentences $[InitialisedTrue(f) \vee InitialisedFalse(f)]$ and $\neg Releases(a, f, t)$ may safely be assumed or added to the axiomatisation. Hence axioms (EC1)–(EC6) simply express the following: (1) Fluents which initially hold, or which have been initiated by an occurrence of an action, continue to hold until an occurrence of an action which terminates them. (2) Fluents which do not initially hold, or which have been terminated by an occurrence of an action, continue not to hold until an occurrence of an action which initiates them.

$$HoldsAt(f,t) \leftarrow [InitialisedTrue(f) \wedge \neg Clipped(0,f,t)] \quad \text{(EC1)}$$

$$\neg HoldsAt(f,t) \leftarrow [InitialisedFalse(f) \wedge \neg Declipped(0,f,t)] \quad \text{(EC2)}$$

$$HoldsAt(f,t_2) \leftarrow [Happens(a,t_1) \wedge Initiates(a,f,t_1) \wedge t_1 < t_2 \wedge \neg Clipped(t_1,f,t_2)] \quad \text{(EC3)}$$

$$\neg HoldsAt(f,t_2) \leftarrow [Happens(a,t_1) \wedge Terminates(a,f,t_1) \wedge t_1 < t_2 \wedge \neg Declipped(t_1,f,t_2)] \quad \text{(EC4)}$$

$$Clipped(t_1,f,t_2) \leftrightarrow \exists a,t[Happens(a,t) \wedge t_1 < t < t_2 \wedge [Terminates(a,f,t) \vee Releases(a,f,t)]] \quad \text{(EC5)}$$

$$Declipped(t_1,f,t_2) \leftrightarrow \exists a,t[Happens(a,t) \wedge t_1 < t < t_2 \wedge [Initiates(a,f,t) \vee Releases(a,f,t)]] \quad \text{(EC6)}$$

This basic event calculus can be extended to deal with continuous change as follows. To respect the convention that actions take effect immediately *after* they occur, it is necessary to axiomatise the mathematical constraint that, at every time-point (including those at which actions occur), the function associated with each parameter is left-hand continuous:

$$LeftContinuous(p,t) \quad \text{(EC7)}$$

To describe instantaneous changes in the values of parameters at times when actions occur, and discontinuities in their corresponding functions of time, the predicates *BreaksTo* and *Breaks* are introduced. Both are minimised. $BreaksTo(A,P,T,R)$ should be read as 'at time T, an occurrence of action A will cause parameter P to instantaneously take on value R'. More precisely, Axiom (EC10) below states that if A also happens at time T, then R is the value of the right-hand limit of P at T. *BreaksTo* is used, for example, to describe the effects of a "scoop" action on the parameter *Level* in the water tank example. $Breaks(A,P,T)$ can be read as 'at time T, action A potentially causes a discontinuity in parameter P'. The following domain-independent axioms make direct use of *BreaksTo* and *Breaks*. Axioms (EC8) and (EC9) can be likened to 'frame axioms' for parameters. Axiom (EC12) states the relationship between *BreaksTo* and *Breaks*, and Axiom (EC12) states that if an action potentially causes a discontinuity in a given parameter, it also potentially causes discontinuities in its higher derivatives.

$$\neg[Happens(a,t) \wedge Breaks(a,p,t)] \rightarrow Continuous(p,t) \quad \text{(EC8)}$$

$$\neg[Happens(a,t) \wedge Breaks(a,\delta(p),t)] \rightarrow Differentiable(p,t) \quad \text{(EC9)}$$

$$[BreaksTo(a,p,t,r) \wedge Happens(a,t)] \rightarrow RightLimit(p,t,r) \quad \text{(EC10)}$$

$$BreaksTo(a,p,t,r) \rightarrow Breaks(a,p,t) \quad \text{(EC11)}$$

$$Breaks(a,p,t) \rightarrow Breaks(a,\delta(p),t) \quad \text{(EC12)}$$

For any given time point T, it is useful to be able to refer to the next point after T at which an action occurs, if there is such a point. Axioms (EC13), (EC14) and (EC15) state that if any action occurs at any time point after T, then the term $Next(T)$ refers to the least such time point. (Such points are somewhat analogous to the "least natural time points" discussed in (Reiter 1996).)

$$t < Next(t) \quad \text{(EC13)}$$

$$[t < t_1 \wedge t_1 < Next(t)] \rightarrow \neg Happens(a,t_1) \quad \text{(EC14)}$$

$$[Happens(a_1,t_1) \wedge t < t_1] \rightarrow \exists a.Happens(a, Next(t)) \quad \text{(EC15)}$$

Finally, the standard mathematical definitions of *Continuous*, *Differentiable*, *LeftContinuous* and *RightLimit* are straightforwardly axiomatised using the function symbols *Value* and δ:

$$Continuous(p,t) \leftrightarrow \forall r \exists t_1 \forall t_2[[|t-t_2| < t_1 \wedge 0 < r] \rightarrow |Value(p,t) - Value(p,t_2)| < r] \quad \text{(A1)}$$

$$Differentiable(p,t) \leftrightarrow \forall r \exists t_1 \forall t_2[[0 < |t-t_2| < t_1 \wedge 0 < r] \rightarrow |(\frac{Value(p,t)-Value(p,t_2)}{t-t_2}) - Value(\delta(p),t)| < r] \quad \text{(A2)}$$

$$LeftContinuous(p,t) \leftrightarrow \forall r \exists t_1 \forall t_2[[t_2 < t \wedge (t-t_2) < t_1 \wedge 0 < r] \rightarrow |Value(p,t) - Value(p,t_2)| < r] \quad \text{(A3)}$$

$$RightLimit(p,t,r) \leftrightarrow \forall r_1 \exists t_1 \forall t_2[[t < t_2 \wedge (t_2-t) < t_1 \wedge 0 < r_1] \rightarrow |Value(p,t_2) - r| < r_1] \quad \text{(A4)}$$

4 DOMAIN-DEPENDENT AXIOMS

4.1 AN AXIOMATISATION OF THE ROBOT EXAMPLE

The robot described in Section 2 can start to move forward or start to turn at any speed, so that two parameterised action symbols are needed, $ChangeSpeed : \mathcal{R} \mapsto \mathcal{A}$ and $StartTurn : \mathcal{R} \mapsto \mathcal{A}$, where, for example, $ChangeSpeed(S)$ signifies the action of "changing the forward speed to S". We can describe the properties that these actions actions initiate and terminate using the parameterised fluent symbols $Moving : \mathcal{R} \mapsto \mathcal{F}$ and $Turning : \mathcal{R} \mapsto \mathcal{F}$, as follows

$$Initiates(ChangeSpeed(r), Moving(r), t) \quad \text{(R1)}$$
$$Initiates(StartTurn(r), Turning(r), t) \quad \text{(R2)}$$
$$Terminates(ChangeSpeed(r_1), Moving(r_2), t) \leftarrow r_1 \neq r_2 \quad \text{(R3)}$$
$$Terminates(StartTurn(r_1), Turning(r_2), t) \leftarrow r_1 \neq r_2 \quad \text{(R4)}$$

The following axioms express the fact that a *ChangeSpeed* action can cause discontinuities in the first derivatives of the parameters *NSP* and *EWP*, whereas a *StartTurn* action can cause discontinuities in the first derivatives of all three parameters in the domain:

$$Breaks(ChangeSpeed(r), \delta(NSP), t) \quad \text{(R5)}$$
$$Breaks(StartTurn(r), \delta(NSP), t) \quad \text{(R6)}$$
$$Breaks(ChangeSpeed(r), \delta(EWP), t) \quad \text{(R7)}$$
$$Breaks(StartTurn(r), \delta(EWP), t) \quad \text{(R8)}$$
$$Breaks(StartTurn(r), \delta(Bearing), t) \quad \text{(R9)}$$

The constraints (1)–(3) of Section 2 can now be axiomatised as follows:

$$Value(\delta(EWP), t) = r.\cos(Value(Bearing, t)) \leftarrow HoldsAt(Moving(r), t) \quad \text{(R10)}$$
$$Value(\delta(NSP), t) = r.\sin(Value(Bearing, t)) \leftarrow HoldsAt(Moving(r), t) \quad \text{(R11)}$$
$$Value(\delta(Bearing), t) = r \leftarrow HoldsAt(Turning(r), t) \quad \text{(R12)}$$

Happens can be used to state that the robot changes speed at time T_1, starts to turn at time T_2, and stops moving whenever it hits the wall:

$$Happens(ChangeSpeed(S^+), T_1) \quad \text{(R13)}$$
$$Happens(StartTurn(A^+), T_2) \quad \text{(R14)}$$
$$[Happens(ChangeSpeed(0), t) \wedge Happens(StartTurn(0), t)] \leftarrow [Value(NSP, t) = D_W \wedge Value(\delta(NSP), t) > 0] \quad \text{(R15)}$$

Finally, axioms are needed stating various initial

conditions, and expressing uniqueness-of-names properties (using the "UNA[...]" notation from (Baker 1991)) for all action, fluent and parameter symbols.

$$[InitialisedTrue(Moving(0)) \wedge InitialisedTrue(Turning(0))] \quad \text{(R16)}$$
$$[InitialisedFalse(Moving(r)) \wedge InitialisedFalse(Turning(r))] \leftarrow r \neq 0 \quad \text{(R17)}$$
$$Value(NSP,0)=0 \wedge Value(EWP,0)=0 \wedge Value(Bearing,0)=0 \quad \text{(R18)}$$
$$\text{UNA}[ChangeSpeed, StartTurn] \quad \text{(R19)}$$
$$\text{UNA}[Moving, Turning] \quad \text{(R20)}$$
$$\text{UNA}[Bearing, NSP, EWP, \delta] \quad \text{(R21)}$$

4.2 AN AXIOMATISATION OF THE WATER TANK EXAMPLE

The following constant symbols will be used to axiomatise the water tanks example. $TurnOnC$, $TurnOnD$ and $Scoop$ of sort $\mathcal{A}$, $OpenC$ and $OpenD$ of sort $\mathcal{F}$, and $Level$, $FlowC$ and $FlowD$ of sort $\mathcal{P}$. The (direct) effects of turning on either tap can be described as follows:

$$Initiates(TurnOnC, OpenC, t) \quad \text{(T1)}$$
$$Initiates(TurnOnD, OpenD, t) \quad \text{(T2)}$$

The action of scooping water from the tank has no effect on the fluents in the domain, but instantaneously effects the parameter $Level$. If the level in the tank is greater than or equal to $\frac{L}{2}$, a $Scoop$ action reduces the level by $\frac{L}{2}$. For the purpose of illustration, we will further suppose that if the level in the tank is less than $\frac{L}{2}$, a scoop action removes all the water from the tank:

$$BreaksTo(Scoop, Level, t, Value(Level,t) - \tfrac{L}{2}) \leftarrow Value(Level,t) \geq \tfrac{L}{2} \quad \text{(T3)}$$
$$BreaksTo(Scoop, Level, t, 0) \leftarrow Value(Level,t) < \tfrac{L}{2} \quad \text{(T4)}$$

As well as causing a discontinuity in $FlowC$, the action $TurnOnC$ causes a discontinuity in the first derivative $Level'$, and, if $TapD$ is open, in the first derivative $FlowD'$. The effects of $TurnOnD$ are analogous:

$$Breaks(TurnOnC, FlowC, t) \quad \text{(T5)}$$
$$Breaks(TurnOnC, \delta(Level), t) \quad \text{(T6)}$$
$$Breaks(TurnOnC, \delta(FlowD), t) \leftarrow HoldsAt(OpenD, t) \quad \text{(T7)}$$
$$Breaks(TurnOnD, FlowD, t) \quad \text{(T8)}$$
$$Breaks(TurnOnD, \delta(Level), t) \quad \text{(T9)}$$
$$Breaks(TurnOnD, \delta(FlowC), t) \leftarrow HoldsAt(OpenC, t) \quad \text{(T10)}$$

Constraints (8)–(12) of Section 2 are axiomatised as follows:

$$Value(\delta(Level), t) = -(Value(FlowC, t) + Value(FlowD, t)) \quad \text{(T11)}$$
$$HoldsAt(OpenC, t) \rightarrow Value(FlowC, t) = K.Value(Level, t) \quad \text{(T12)}$$
$$\neg HoldsAt(OpenC, t) \rightarrow Value(FlowC, t) = 0 \quad \text{(T13)}$$
$$HoldsAt(OpenD, t) \rightarrow Value(FlowD, t) = K.Value(Level, t) \quad \text{(T14)}$$
$$\neg HoldsAt(OpenD, t) \rightarrow Value(FlowD, t) = 0 \quad \text{(T15)}$$

The action occurrences are:

$$Happens(Scoop, T_1) \quad \text{(T16)}$$
$$Happens(TurnOnC, T_1) \quad \text{(T17)}$$
$$Happens(TurnOnD, T_2) \quad \text{(T18)}$$

Finally, the initial conditions and uniqueness-of-names axioms are:

$$InitialisedFalse(OpenC) \quad \text{(T19)}$$
$$InitialisedFalse(OpenD) \quad \text{(T20)}$$
$$Value(Level, 0) = L \quad \text{(T21)}$$
$$\text{UNA}[TurnOnC, TurnOnD, Scoop] \quad \text{(T22)}$$
$$\text{UNA}[OpenC, OpenD] \quad \text{(T23)}$$
$$\text{UNA}[Level, FlowC, FlowD, \delta] \quad \text{(T24)}$$

5 THE CIRCUMSCRIPTION POLICY CIRC_{CEC}

By themselves, the axioms in Section 4.1 are not sufficient to infer the robot's trajectory. In general, a default reasoning mechanism will also be required which models various default assumptions about such domains. The circumscription policy used here is inspired by a solution to the frame problem described by Kartha and Lifschitz (1995), which is related to Sandewall's idea of *filter preferential entailment* (Sandewall 1989b). This has been adapted for use with the event calculus, and extended so that it also models the assumptions that by default a given action does not occur at a given time point, and that by default a given action occurrence does not result in a discontinuity for a given parameter. Given a collection of domain-dependent axioms $\mathcal{D}$ similar to those in the previous

section, the circumscription policy is:

$$\begin{aligned} &CIRC[Nar(\mathcal{D})\ ;\ Happens] \\ &\wedge CIRC[Eff(\mathcal{D})\ ;\ Initiates, Terminates, Releases] \\ &\wedge CIRC[(Inst(\mathcal{D}) \wedge (\text{EC11}) \wedge (\text{EC12}))\ ; \\ &\qquad\qquad Breaks\ ;\ BreaksTo] \\ &\wedge Con(\mathcal{D}) \wedge Una(\mathcal{D}) \wedge [(\text{EC1}) \wedge \ldots \wedge (\text{EC15})] \end{aligned}$$

We will abbreviate this to $CIRC_{CEC}[\mathcal{D}]$. In $CIRC_{CEC}[\mathcal{D}]$, the term "$Nar(\mathcal{D})$" stands for (the conjunction of) those domain-specific axioms describing the "narrative" (e.g. *Happens* facts and statements about the initial values of fluents or parameters), "$Eff(\mathcal{D})$" stands for those axioms describing the effects of actions on fluents (using *Initiates*, *Terminates* and *Releases*), "$Inst(\mathcal{D})$" stands for those axioms describing the instantaneous effects of actions on parameters (using *Breaks* and *BreaksTo*), "$Con(\mathcal{D})$" stands for axioms describing mathematical constraints between parameters during different circumstances (e.g. when a tap is open, its flow is proportional to the water level in the tank), and "$Una(\mathcal{D})$" stands for the uniqueness-of-names axioms. So, if $\mathcal{D}_R$ is the set of axioms describing the robot,

$$\begin{aligned} Nar(\mathcal{D}_R) &= [(\text{R13}) \wedge \ldots \wedge (\text{R18})] \\ Eff(\mathcal{D}_R) &= [(\text{R1}) \wedge \ldots \wedge (\text{R4})] \\ Inst(\mathcal{D}_R) &= [(\text{R5}) \wedge \ldots \wedge (\text{R9})] \\ Con(\mathcal{D}_R) &= [(\text{R10}) \wedge \ldots \wedge (\text{R12})] \\ Una(\mathcal{D}_R) &= [(\text{R19}) \wedge \ldots \wedge (\text{R21})] \end{aligned}$$

and if $\mathcal{D}_T$ is the set of axioms describing the water tank example,

$$\begin{aligned} Nar(\mathcal{D}_T) &= [(\text{T16}) \wedge \ldots \wedge (\text{T21})] \\ Eff(\mathcal{D}_T) &= [(\text{T1}) \wedge (\text{T2})] \\ Inst(\mathcal{D}_T) &= [(\text{T3}) \wedge \ldots \wedge (\text{T10})] \\ Con(\mathcal{D}_T) &= [(\text{T11}) \wedge \ldots \wedge (\text{T15})] \\ Una(\mathcal{D}_T) &= [(\text{T22}) \wedge \ldots \wedge (\text{T24})] \end{aligned}$$

Whenever $Nar(\mathcal{D})$, $Eff(\mathcal{D})$ and $Inst(\mathcal{D})$ are of a certain general form, we can prove general properties of $CIRC_{CEC}[\mathcal{D}]$ which allow its consequences to be computed using classical deduction. The three propositions below are applicable to a wide class of domains which includes both $\mathcal{D}_R$ and $\mathcal{D}_T$. (Strictly speaking, to fit the conditions of the propositions, some of the domain-dependent axioms in $\mathcal{D}_R$ and $\mathcal{D}_T$ must be re-written in a slightly different form. For example, (T3) is re-written as

$$\begin{aligned} &BreaksTo(a,p,t,r) \leftarrow \qquad (\text{T3}') \\ &\quad [Value(Level,t) \geq \tfrac{L}{2} \wedge a = Scoop \\ &\quad \wedge p = Level \wedge r = Value(Level,t) - \tfrac{L}{2}] \end{aligned}$$

and similar syntactic transformations are applied to the other clauses partially defining *Initiates*, *Terminates*, *Happens*, *Breaks* or *BreaksTo*.)

Proposition 1 Let S be the conjunction of (EC11) and (EC12) with the following sentences:

$$\begin{aligned} &Breaks(a,p,t) \leftarrow \Phi_1(a,p,t) && (\text{S1}) \\ &\qquad\vdots && \vdots \\ &Breaks(a,p,t) \leftarrow \Phi_k(a,p,t) && (\text{Sk}) \\ &BreaksTo(a,p,t,r) \leftarrow \Phi_{k+1}(a,p,t,r) && (\text{Sk+1}) \\ &\qquad\vdots && \vdots \\ &BreaksTo(a,p,t,r) \leftarrow \Phi_m(a,p,t,r) && (\text{Sm}) \end{aligned}$$

where a, p and t are the only variables which appear free in the formulae $\Phi_1(a,p,t), \ldots, \Phi_k(a,p,t)$, where a, p, t and r are the only variables which appear free in the formulae $\Phi_{k+1}(a,p,t,r)$, $\ldots, \Phi_m(a,p,t,r)$, and where none of $\Phi_1(a,p,t), \ldots, \Phi_k(a,p,t)$ or $\Phi_{k+1}(a,p,t,r), \ldots, \Phi_m(a,p,t,r)$ mention the predicates *Breaks* or *BreaksTo*. Then $CIRC[S\ ;\ Breaks\ ;\ BreaksTo]$ entails the following sentence S_{comp}:

$$\begin{aligned} &Breaks(a,p,t) \leftrightarrow \\ &\quad [\exists p_1 [p = \delta(p_1) \wedge Breaks(a,p_1,t)] \vee \\ &\quad \Phi_1(a,p,t) \vee \ldots \vee \Phi_k(a,p,t) \vee \\ &\quad \exists r [\Phi_{k+1}(a,p,t,r) \vee \ldots \vee \Phi_m(a,p,t,r)]] \end{aligned}$$

Proof: (*Notation:* Let $\overline{x}$ be the tuple of free variables in the formula $\Phi(\overline{x})$. Then in the following proof, $\overline{\Phi}$ refers to the *predicate expression* $\lambda\overline{x}.\Phi(\overline{x})$ (see (Lifschitz 1995) for definition). Hence, given a model M and an appropriately sorted tuple of domain objects $\overline{\chi}$, the statement $\overline{\chi} \in M[\![\overline{\Phi}]\!]$ signifies that $M, v \Vdash \overline{\Phi}(\overline{x})$ for all variable assignments v such that $v(\overline{x}) = \overline{\chi}$.)

The if half of S_{comp} follows directly from S. It remains to prove the only-if half. Suppose there is some model M of S which does not satisfy the only-if half of S_{comp}. Then there must be some $\langle \alpha, \rho, \tau \rangle \in M[\![Breaks]\!]$ such that:

(i) there is no ρ' such that $\rho = M[\![\delta]\!](\rho')$ and $\langle \alpha, \rho', \tau \rangle \in M[\![Breaks]\!]$

(ii) for all $i \leq k$, $\langle \alpha, \rho, \tau \rangle \notin M[\![\overline{\Phi_i}]\!]$

(iii) for all $k < i \leq m$, there is no π such that $\langle \alpha, \rho, \tau, \pi \rangle \in M[\![\overline{\Phi_i}]\!]$

Furthermore, since M is a model of S, it follows that for each parameter ρ'' such that $\rho = M[\![\delta]\!]^n(\rho'')$ for some $n \geq 1$:

(iv) $\langle \alpha, \rho'', \tau \rangle \notin M[\![Breaks]\!]$

(v) for all $i \leq k$, $\langle \alpha, \rho'', \tau \rangle \notin M[\![\overline{\Phi_i}]\!]$

(vi) for all $k < i \leq m$, there is no π such that $\langle \alpha, \rho'', \tau, \pi \rangle \in M[\![\overline{\Phi_i}]\!]$

(Otherwise, we could put $\rho' = M[\![\delta]\!]^{n-1}(\rho'')$ and (i) would not be satisfied.) Hence we can construct a smaller model than M by removing $\langle \alpha, \rho, \tau \rangle$ from $M[\![Breaks]\!]$. In order to satisfy (E13), this necessitates the additional removal of all tuples of the form $\langle \alpha, \rho, \tau, \pi \rangle$ from $M[\![BreaksTo]\!]$. More precisely, let M' be an interpretation obtained from M in the following way:

- M' agrees with M on the interpretation of all predicate, constant and function symbols except *Breaks* and *BreaksTo*.
- $\langle \alpha', \rho', \tau' \rangle \in M'[\![Breaks]\!]$ if and only if $\langle \alpha', \rho', \tau' \rangle \in M[\![Breaks]\!]$ and $\langle \alpha', \rho', \tau' \rangle \neq \langle \alpha, \rho, \tau \rangle$.
- $\langle \alpha', \rho', \tau', \pi \rangle \in M'[\![BreaksTo]\!]$ if and only if $\langle \alpha', \rho', \tau', \pi \rangle \in M[\![BreaksTo]\!]$ and $\langle \alpha', \rho', \tau' \rangle \neq \langle \alpha, \rho, \tau \rangle$.

It is easily verified that M' is also a model of S. Since $M'[\![Breaks]\!]$ is a strict subset of $M[\![Breaks]\!]$, M' is preferable to M according to the circumscription policy. Therefore M cannot be a model of $CIRC[S\ ;\ Breaks\ ;\ BreaksTo]$, and the proposition holds. □

Proposition 2 Let S be the conjunction of the following sentences:

$$\begin{array}{lll} \mathit{Happens}(a,t) & \leftarrow & \Phi_1(a,t) \quad \text{(S1)} \\ \vdots & & \vdots \\ \mathit{Happens}(a,t) & \leftarrow & \Phi_n(a,t) \quad \text{(Sn)} \end{array}$$

where a and t are the only variables which (possibly) appear free in the formulae $\Phi_1(a,t), \ldots, \Phi_n(a,t)$, and where none of $\Phi_1(a,t), \ldots, \Phi_n(a,t)$ mention the predicate *Happens*. Then $CIRC[S\ ;\ Happens]$ entails the following sentence:

$$\mathit{Happens}(a,t) \leftrightarrow [\Phi_1(a,t) \vee \ldots \vee \Phi_n(a,t)]$$

Proof: The proposition follows directly from Proposition 3.1.1 in (Lifschitz 1995). □

Proposition 3 Let S be the conjunction of the following sentences:

$$\begin{array}{lll} \mathit{Initiates}(a,f,t) & \leftarrow & \Phi_1(a,f,t) \quad \text{(S1)} \\ \vdots & & \vdots \\ \mathit{Initiates}(a,f,t) & \leftarrow & \Phi_k(a,f,t) \quad \text{(Sk)} \\ \mathit{Terminates}(a,f,t) & \leftarrow & \Phi_{k+1}(a,f,t) \quad \text{(Sk+1)} \\ \vdots & & \vdots \\ \mathit{Terminates}(a,f,t) & \leftarrow & \Phi_m(a,f,t) \quad \text{(Sm)} \\ \mathit{Releases}(a,f,t) & \leftarrow & \Phi_{m+1}(a,f,t) \quad \text{(Sm+1)} \\ \vdots & & \vdots \\ \mathit{Releases}(a,f,t) & \leftarrow & \Phi_n(a,f,t) \quad \text{(Sn)} \end{array}$$

where a, f and t are the only variables which (possibly) appear free in the formulae $\Phi_1(a,f,t), \ldots, \Phi_n(a,f,t)$, and where none of $\Phi_1(a,f,t), \ldots, \Phi_n(a,f,t)$ mention the predicates *Initiates*, *Terminates* or *Releases*.

Then $CIRC[S\ ;\ \mathit{Initiates}, \mathit{Terminates}, \mathit{Releases}]$ entails the following three sentences:

$$\mathit{Initiates}(a,f,t) \leftrightarrow [\Phi_1(a,f,t) \vee \ldots \vee \Phi_k(a,f,t)]$$

$$\mathit{Terminates}(a,f,t) \leftrightarrow [\Phi_{k+1}(a,f,t) \vee \ldots \vee \Phi_m(a,f,t)]$$

$$\mathit{Releases}(a,f,t) \leftrightarrow [\Phi_{m+1}(a,f,t) \vee \ldots \vee \Phi_n(a,f,t)]$$

Proof: The proposition follows directly from Propositions 3.1.1 and 7.1.1 in (Lifschitz 1995). □

Using these results, it is not hard to generate classical derivations of sentences of the form $\mathit{HoldsAt}(F,T)$, $\mathit{Happens}(A,T)$ and $\mathit{Value}(P,T)=R$ as required. For example, an outline derivation of the sentence

$$\mathit{Happens}(\mathit{ChangeSpeed}(0), T_2 + \tfrac{1}{A^+}\arccos(1 - \tfrac{A^+ \cdot D_W}{S^+}))$$

from $CIRC_{CEC}[\mathcal{D}_R]$ is given in Appendix B of the longer version of this paper (Miller & Shanahan 1996).

6 SUMMARY AND DISCUSSION

A logical formalism for representing both discrete and continuous change has been presented which overcomes a number of drawbacks in existing logic-based formalisms. It permits the use of arbitrary formulae of the differential calculus without giving rise to anomalous models. In addition, it incorporates a solution to the frame problem which is robust in the presence of narrative, concurrent actions, non-deterministic actions, and domain constraints.

It is important to incorporate the notion of a derivative function in any comprehensive formalism for mod-

elling domains with continuous change. Differential calculus is the primary tool for mathematical modelling in mainstream science and engineering. When a reasonably complex dynamic system is represented as a set of differential equations it is often not possible to obtain an analytical solution. Instead, numerical methods may be used, and the present formalism is a step towards integrating such computational techniques with systems for automated reasoning about actions. Furthermore, the information about the continuously varying aspects of a domain may be incomplete, in which case it may be more appropriate to use computational methods from Qualitative Reasoning (Weld 1990) (Kuipers 1994). Again, the notion of a derivative function is fundamental to the semantics of such systems.

The mechanisms incorporated in the formalism for reasoning about boundary conditions, using the predicates *Breaks* and *BreaksTo*, are similar to those introduced in (Miller 1996). However, whereas the discussion in (Miller 1996) was restricted to a particular case study, Propositions 1, 2 and 3 have allowed the effects of the circumscription policy $CIRC_{CEC}$ to be characterised for a wide class of domains. Moreover, they allow the use of standard, first-order proof-theoretic techniques (as opposed to the model-theoretic arguments of (Miller 1996), (Sandewall 1989a) or (Sandewall 1989b)) to ascertain logical consequences of a given domain description. Indeed, the overall structure of the example derivation given in (Miller & Shanahan 1996) points towards a particular approach to designing algorithms for temporal projection, which would be sound with respect to the logical specification presented here. Such algorithms would compute forward in time in alternating "steps", each step being either a single time point identified by the *Next* function (see Section 3), or an open interval between such time points.

A great deal of work has already been done on algorithms which perform this kind of computation in the Qualitative Reasoning community (Weld 1990) (Kuipers 1994). In the terminology of qualitative reasoning, time points identified by the *Next* function may often coincide with *distinguished* or *landmark time-points* with respect to some continuously varying parameter. Algorithms for reasoning about landmark time-points and the associated landmark values of particular parameters are embedded in, for example, QSIM (Kuipers 1986), and in systems based on Qualitative Process Theory (Forbus 1984). Furthermore, qualitative process theory incorporates the ability to handle actions (Forbus 1989). Although some effort has been made to reconcile logic-based work in Reasoning about Action with that in Qualitative Reasoning (Crawford & Etherington 1992) (van Belleghem *et al* 1994), the two fields have yet to be properly integrated. Attempts to axiomatise qualitative reasoning are valuable here (Davis 1992), and it is hoped that the present paper can serve to further work in this vein.

Of course, the emphasis in Qualitative Reasoning is on reasoning with incomplete or qualitative information about relationships between parameters. In the present formalism, such information would manifest itself in the use of the inequality predicate and existentially quantified numerical variables in constraints between parameters, or in the use of appropriately defined "qualitative predicates". This is discussed further in Appendix A. Domain dependent qualitative constraints would appear in axioms analogous to (R10)–(R12) in the "*Con*" part of the theory (see Section 5). The fact that $Con(\mathcal{D})$ is outside the scope of any circumscription indicates that the minimisation policy is equally applicable to either qualitative or quantitative domain descriptions. It is therefore hoped that, as well as integrating the notions of discrete and continuous change, the present formalism is a step towards providing a unifying conceptual framework for qualitative, semi-qualitative and quantitative reasoning about continuous change. But further work needs to be done in order to substantiate this claim.

Acknowledgement

This research was funded by the U.K. Engineering and Physical Sciences Research Council (EPSRC).

References

A. Baker (1991), *Nonmonotonic Reasoning in the Framework of the Situation Calculus*, A.I. vol. 49, page 5, Elsevier Science Publishers.

J. Crawford and D. Etherington (1992), *Formalizing Reasoning about Change: A Qualitative Reasoning Approach*, Proceedings AAAI'92, pages 577-583.

E. Davis (1992a), *Axiomatising Qualitative Process Theory*, Proceedings KR'92, Morgan Kaufmann, pages 177-188.

E. Davis (1992b), *Infinite Loops in Finite Time: Some Observations*, Proceedings KR'92, Morgan Kaufmann.

K. Forbus (1984), *Qualitative Process Theory*, in Artificial Intelligence 24, reprinted in Weld and de Kleer (eds.), Readings in Qualitative Reasoning about Physical Systems, Morgan Kaufmann (1990), 1984.

K. Forbus (1989), *Introducing Actions into Qualitative Simulation*, Proceedings IJCAI'89, pages 1273-1278.

C. Herrmann and M. Thielscher (1996), *Reasoning about Continuous Processes*, Proceedings AAAI'96.

G.N. Kartha and V. Lifschitz (1994), *Actions with Indirect Effects (Preliminary Report)*, Proceedings KR'94, pages 341-350.

G.N. Kartha and V. Lifschitz (1995), *A Simple Formalization of Actions Using Circumscription*, Proceedings IJCAI'95, pages 1970-1975.

R. Kowalski and M. Sergot (1986), *A Logic-Based Calculus of Events*, New Generation Computing, vol. 4, page 267.

B. Kuipers (1986), *Qualitative Simulation*, A.I. vol. 29, pages 289-338, Elsevier Science Publishers.

B. Kuipers (1994), *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*, MIT Press.

V. Lifschitz (1995), *Circumscription*, in Handbook of Logic in Artificial Intelligence, ed.s D. Gabbay, C. Hogger and J.A. Robinson, Oxford University Press, pages 297-352.

R. Miller (1995), *Situation Calculus Specifications for Event Calculus Logic Programs*, in Proceedings of the Third International Conference on Logic Programming and Non-monotonic Reasoning, Lexington, KY, USA, pub. Springer Verlag.

R. Miller (1996), *A Case Study in Reasoning about Actions and Continuous Change*, Proceedings ECAI'96, pub. John Wiley & Sons, Ltd.

R. Miller and M. Shanahan (1994), *Narratives in the Situation Calculus*, in Journal of Logic and Computation, Special Issue on Actions and Processes, vol. 4 no. 5, Oxford University Press.

R. Miller and M. Shanahan (1996), *Reasoning about Discontinuities in the Event Calculus (Extended Version)*, `http://www-lp.doc.ic.ac.uk/UserPages/staff/rsm/abstract12.html`.

J. Pinto (1994), *Temporal Reasoning in the Situation Calculus*, PhD. Thesis, University of Toronto.

M. Rayner (1991), *On the Applicability of Nonmonotonic Logic to Formal Reasoning in Continuous Time*, A.I. vol. 49, pages 345-360, Elsevier Science.

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 Honour of John McCarthy, ed. V. Lifschitz, Academic Press, page 418.

R. Reiter (1996), *Natural Actions, Concurrency and Continuous Time in the Situation Calculus*, Proceedings KR'96 (this proceedings), Morgan Kaufmann.

E. Sandewall (1989a), *Combining Logic and Differential Equations for Describing Real World Systems*, Proceedings KR'89, Morgan Kaufman.

E. Sandewall (1989b), *Filter Preferential Entailment for the Logic of Action in Almost Continuous Worlds*, Proceedings IJCAI'89, pages 894-899.

E. Sandewall (1994), *Features and Fluents*, Oxford University Press.

M. Shanahan (1990), *Representing Continuous Change in the Event Calculus*, Proceedings ECAI'90, pages 598-603.

M. Shanahan (1995), *A Circumscriptive Calculus of Events*, A.I. vol. 77, pages 249-284, Elsevier Science.

M. Shanahan (1996), *Robotics and the Common Sense Informatic Situation*, in Proceedings ECAI'96, pub. John Wiley & Sons, Ltd.

M. Shanahan (1997), *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia*, MIT Press, (to appear).

Y. Shoham (1988), *Reasoning About Change: Time and Causation from the Standpoint of Artificial Intelligence*, MIT Press.

K. Van Belleghem, M. Deneker and D. De Schreye (1994), *Representing Continuous Change in the Abductive Event Calculus*, in Proceedings 1994 International Conference on Logic Programming, ed. P. Van Hentenrijck, pages 225-240.

D. Weld and J. de Kleer (editors) (1990), *Qualitative Reasoning about Physical Systems*, Morgan Kaufmann.

APPENDIX A – REPRESENTING QUALITATVE INFORMATION ABOUT PARAMETER BEHAVIOUR

This appendix outlines some preliminary ideas for incorporating qualitative information about parameter behaviour in domain descriptions, along the lines of (Kuipers 1986) and (Kuipers 1994). The discussion here is speculative – a more thorough investigation is needed to fully integrate the notions and ontologies of QR with those of action based logical formalisms.

Kuipers lists seven qualitative relationships between parameters as being particularly important as regards the qualitative description of a system's behaviour. These relationships are *add*, *mult*, *minus*, *d/dt*, *con-*

stant, M^+ (monotonically increasing) and M^- (monotonically decreasing). These notions can be included in the present framework by appropriately extending the axiom set (A1)-(A4):

$$Add(p_1,p_2,p_3,t) \leftrightarrow \quad (Q1)$$
$$Value(p_1,t)+Value(p_2,t)=Value(p_3,t)$$

$$Mult(p_1,p_2,p_3,t) \leftrightarrow \quad (Q2)$$
$$Value(p_1,t).Value(p_2,t)=Value(p_3,t)$$

$$Minus(p_1,p_2,t) \leftrightarrow \quad (Q3)$$
$$Value(p_1,t)=-Value(p_2,t)$$

$$Derivative(p_1,p_2,t) \leftrightarrow \quad (Q4)$$
$$Value(\delta(p_1),t)=Value(p_2,t)$$

$$Constant(p,t) \leftrightarrow Value(\delta(p),t)=0 \quad (Q5)$$

$$MonInc(p_1,p_2,t) \leftrightarrow \quad (Q6)$$
$$[[Value(\delta(p_1),t)>0 \leftrightarrow Value(\delta(p_2),t)>0] \wedge$$
$$[Value(\delta(p_1),t)<0 \leftrightarrow Value(\delta(p_2),t)<0]]$$

$$MonDec(p_1,p_2,t) \leftrightarrow \quad (Q7)$$
$$[[Value(\delta(p_1),t)>0 \leftrightarrow Value(\delta(p_2),t)<0] \wedge$$
$$[Value(\delta(p_1),t)<0 \leftrightarrow Value(\delta(p_2),t)>0]]$$

With such extra definitions, it is easy to include qualitative constraints in domain descriptions. For example, in the water tank example we might replace axiom (T12) with the sentence

$$HoldsAt(OpenC,t) \rightarrow MonInc(FlowC,Level,t)$$

to reflect the fact that, although we know that the flow through tap C decreases as the water level decreases, we do not know the exact mathematical relationship between the two parameters in this circumstance.

Another notion central to QR is that of a *landmark value* (w.r.t. a particular parameter). A landmark value, or simply "landmark", is a particular "qualitatively important" value for some parameter, typically where the first derivative changes sign. Symbolic names may be used for landmarks, since their exact numerical values may be unknown. In the present context, landmarks can be represented by extra constant or function symbols, along with suitable sets of ordering declarations. For example, qualitatively important values for the parameter *Level* in the water tank example are 0 and the top of the tank. Hence the "quantity space" for *Level* can be described by the sentence

$$0 < Top$$

where *Top* is an extra constant symbol of sort $\mathcal{R}$. In the terminology of QSIM, this quantity space gives rise to 15 possible qualitative values for the parameter *Level*, each of the form $\langle qmag, qdir\rangle$, where *qmag* is chosen from the set $\{(-\infty,0),0,(0,Top),Top,(Top,\infty)\}$, and *qdir* is chosen from the set $\{Inc, Std, Dec\}$ ("increasing", "steady" or "decreasing"). Qualitative values can be represented as fluents within the present framework by incorporating (6) domain independent axioms of the following form:

$$HoldsAt(QV(p,\langle(r_1,r_2),Inc\rangle),t) \leftrightarrow \quad (Q8)$$
$$[r_1<Value(p,t)<r_2 \wedge Value(\delta(p),t)>0]$$

$$HoldsAt(QV(p,\langle r,Inc\rangle),t) \leftrightarrow \quad (Q9)$$
$$[Value(p,t)=r \wedge Value(\delta(p),t)>0]$$

The fluent $QV(Level,\langle(0,Top),Inc\rangle)$ is not included in the "frame". Conceptually, its truth value at any instant of time is instead determined by the values of *Level* and $\delta(Level)$. With such notation, axiom (T21) stating the initial value of *Level* could be replaced with an assertion such as

$$HoldsAt(QV(Level,\langle(0,Top),Std\rangle),0)$$

or

$$\exists x.HoldsAt(QV(Level,\langle(0,Top),x\rangle),0)$$

Similarly, we could describe when triggered events such as as an "overflow" occur with axioms of the form

$$Happens(Overflow,t) \leftarrow$$
$$HoldsAt(QV(Level,\langle Top,Inc\rangle),t)$$

Computationally, for domains involving both discrete and continuous change a hybrid system seems feasible, in which an event calculus style mechanism (perhaps using logic programming) determines the instantaneous changes in system configuration at times (identified by the *Next* function) when actions (events) occur, and where a QSIM style algorithm determines the possible evolutions of the continuously varying aspects of the domain between such time points. The interaction between these two components would be in both directions - actions could impose new and remove existing qualitative constraints between parameters, and achievement of landmark values by particular parameters could trigger particular action occurrences.

Ramification

Determining Ramifications in the Situation Calculus

Enrico Giunchiglia
Mechanized Reasoning Group
DIST - Università di Genova
Via Opera Pia 13, 16145 Genova, Italy
Email: enrico@dist.unige.it

Abstract

We focus on the ramification problem in the setting of the situation calculus. Our analysis is restricted to theories (or domains) consisting of effect propositions and (ramification) constraints. For these domains, we first identify the class of models we are interested in characterizing. In the case of finite domains, we provide a set of second order formulas defining the models we are interested in, and show how sometimes such formulas can be reduced to a first order theory. When a domain (in the situation calculus) "corresponds" to a domain description in the high level action description language $\mathcal{AR}$, we show that these two formalizations yield the same conclusions.

1 INTRODUCTION

The "frame problem" —how to represent succinctly what remains unchanged as a result of performing an action— has long been a challenge for many researchers in Artificial Intelligence. Several solutions have been proposed (see e.g. [Pednault, 1989; Baker, 1991; Reiter, 1991; Gelfond and Lifschitz, 1992; Kartha and Lifschitz, 1995]), some of these proposals have been formally compared [Kartha, 1993], and there is the feeling that at least for domains without ramification constraints the frame problem is solved.

In this paper we address the frame problem in presence of ramification constraints. We also assume that actions may not be executable, e.g. because their effects violate some constraint. Our analysis is restricted to domains consisting of effect propositions and ramification constraints.

The path we follow is the following. We first identify the "intended models" we want to characterize (Section 2). We thus introduce the notion of model which is "minimally abnormal" and in which "actions are maximally executable". Second, (Section 3), we try to provide a syntactic characterization for these classes of models. In the case of finite domains, we provide two second order formulas each defining one of the classes of models we are interested in. As a third step, we see under which circumstances we can reduce the second order formulas to first order equivalent ones (Section 4). Finally (Section 5), we show that the class of intended models of a domain in the situation calculus entails exactly the same conclusions entailed by the corresponding domain description in a high level action language like $\mathcal{A}$ [Gelfond and Lifschitz, 1992], $\mathcal{AR}_0$ [Kartha and Lifschitz, 1994] or $\mathcal{AR}$ [Giunchiglia *et al.*, 1996].

The novelties and motivations of each of the above four steps are the following. Definitions similar to our definition of "minimally abnormal model" have already been proposed in the literature [Lin and Shoham, 1991; Lin and Reiter, 1994]. One difference is that Lin, Reiter and Shoham restrict their attention to languages with propositional fluents. On the other hand, nonpropositional fluents allow for more natural formulations of many scenarios and their treatment is not as straightforward as it could be expected (see e.g. [Giunchiglia *et al.*, 1995]). The definition of model with maximally executable actions (as far as we know) has never been proposed in the literature. Singling out these two characterizations is important for the goal of building "provably correct theories of actions" in the spirit of [Lin and Shoham, 1991].

The closest work to our second order characterization of the class of minimally abnormal models seems [Kartha and Lifschitz, 1995]. Both approaches are based on the "theory update" view of [Winslett, 1988]. One difference is that the set of domains we consider is wider. In particular, we allow for

nonpropositional fluents and arbitrary formulas (and not just atomic) in the post-condition part of effect propositions. Another difference is that in [Kartha and Lifschitz, 1995], a Nested Abnormality Theory (NAT) [Lifschitz, 1995] is used, while we propose a direct encoding in second order logic. Though the former seems more elegant, it requires at least one more second order quantifier (which have to be eliminated if we want to find a first order reduction). Another approach for such a characterization, is Baker's [Baker, 1991] (but see also [Kartha, 1994; Kartha, 1996]). The biggest drawback that we see in Baker's approach is the presence of higher order circumscription for the existence of situations. For these reasons, the approaches of [Kartha and Lifschitz, 1995] and [Baker, 1991] do not seem to allow for first order reductions as straightforward as ours. The second order characterization of the class of models in which actions are maximally executable (as far as we know) has never been proposed for the set of domains we consider.

The reduction to first order seems to be of particular interest from the perspective of implementing these theories. Recently, there seems to be a growing interest in implementing nonmonotonic reasoning systems (see e.g. [IJCAI, 1995; ECAI, 1996]). As far as circumscription is involved, a lot of efforts are put in mechanizing systems for the elimination of second order quantifiers (see for example [Gabbay and Ohlbach, 1992; Kartha and Lifschitz, 1995; Doherty *et al.*,]). Having a direct encoding in first order logic eliminates the need for this complex step. Furthermore, the set of domains for which the reduction is possible is fairly big. For example, it includes the domains for which Lin and Reiter [Reiter, 1991; Lin and Reiter, 1994] propose a simple solution for the characterization of a class of minimally abnormal models. For such a characterization, Lin and Reiter impose certain consistency conditions and a second order formula ensuring a "tree structure" for the universe of situations. Our solution does need neither of them. Furthermore the reduction to first order is possible for any domain without terms denoting objects, e.g. the domain corresponding to Example 3 of [Kartha and Lifschitz, 1995]. That example involves a "ternary state constraint" (in the sense of Section 3.1 of [Pinto, 1995]). Thus for that domain it is not possible to compute the successor state axiom following the methods developed in [Pinto, 1995].

Finally, the last step is interesting in two ways. First, it supports our definition of "intended model". When for a domain D there is a corresponding domain description $\mathcal{D}$ in $\mathcal{AR}$, the consequences of $\mathcal{D}$ are exactly those entailed by the class of intended models of D. Second, we give a sound and complete translation of *any* domain description in $\mathcal{AR}$ into first order logic. This translation is also applicable to domain descriptions which do not have a counter-part in the domains we consider in Section 3. For example, it is applicable to domain descriptions in which some of the fluents are not inertial (or not in the "frame" [Lifschitz, 1990]) or can nondeterministically change their values.

2 A SEMANTIC SOLUTION

In this paper, all free variables in a formula are considered implicitly universally quantified from the outside. We also assume that bound variables are renamed in such a way that any two quantifiers one in the scope of the other bind distinct variables. By $\phi(z)$ we indicate a formula whose free variables are in z.

We are using a many sorted language with sorts for situations (*situation*), actions (*action*), objects (*object*) and values (*value*). We denote the variables for situations, actions, values and tuples of objects with $s, s', s'', s_1, \ldots, a, a_1, \ldots, v, w, v_1, w_1, \ldots$ and $y, x, x_1, x_2, \ldots$ respectively. As usual, we assume that the sorts be pairwise disjoint and that for each sort S, an equality symbol $=_S$ belongs to the language. We usually drop the S and simply write $=$ for $=_S$.

The only symbols (other than equality) involving the sort *situation* are:

- a set of fluent symbols, denoted with $F, F_1, F_2, \ldots$ Formally, a *fluent symbol* F is a function symbol F: $object^n \times situation \rightarrow value$,
- the binary function do: $action \times situation \rightarrow situation$,
- the constants $\perp$ and S_0, for the undefined and initial situation respectively.

The domains we consider are characterized by a set of

1. *ramification constraints*, (or *constraints*) of the form
$$\Upsilon(s) \tag{1}$$
where s is the only situation term occurring in Υ, and
2. *effect propositions*, of the form
$$do(a, s) = s' \wedge s' \neq \perp \supset \forall x(\Phi(x, a, s) \supset \Psi(x, a, s')) \tag{2}$$
where s [resp. s'] is the only situation term occurring in Φ [resp. Ψ].

Formally, a *domain* consists of the formula

$$S_0 \neq \bot, \tag{3}$$

effect propositions and constraints. Equation (3) is required for ruling out from the class of models those structures whose situation universe is a singleton (and thus in which all the ground situation terms are equal to the undefined situation).

A domain D is *propositional* if the constraint

$$\neg v = \mathsf{T} \equiv v = \mathsf{F} \tag{4}$$

belongs to D. When considering propositional domains, we abbreviate any atomic formula

$$F(\chi, \xi) = \mathsf{T}.$$

with $F(\chi, \xi)$. In writing domains, we usually omit (3) and, in the cases of propositional domains, we omit also (4). With these conventions, the "murder mystery" [Baker, 1991] can be formalized by the propositional domain:

$$\begin{array}{l} do(a,s) = s' \wedge s' \neq \bot \supset (a = Load \supset Loaded(s')), \\ do(a,s) = s' \wedge s' \neq \bot \supset \\ \qquad (a = Shoot \wedge Loaded(s) \supset \neg Alive(s')), \\ do(a,s) = s' \wedge s' \neq \bot \supset (a = Shoot \supset \neg Loaded(s')), \\ Walking(s) \supset Alive(s), \\ Load \neq Wait, Load \neq Shoot, Wait \neq Shoot. \end{array} \tag{5}$$

In the above domain, the first three formulas are effect propositions. All the other formulas are constraints.

Given (5), we are interested in characterizing the models satisfying, e.g.

$$\neg Alive(do(Shoot, do(Wait, do(Load, S_0)))) \tag{6}$$

(i.e. the person is not alive after we load the gun, wait and then shoot), and

$$do(Shoot, do(Wait, do(Load, S_0))) \neq \bot \tag{7}$$

(i.e. it is possible to load the gun, wait and finally shoot).

The intended models of a domain are those which are "minimally abnormal" and in which "actions are maximally executable", in the sense made precise by the following two definitions.

In the following, we indicate the interpretation of a symbol K in a structure M with $M[\![K]\!]$. We also extend the signature of the language by adding to it, for every element k in a universe of M, a new object constant k^* of the appropriate sort. The interpretation of the new symbols is defined by $M[\![k^*]\!] = k$.

Definition 1 *A model M of a domain D is minimally abnormal if there does not exist a model M' of D such that*

1. *M and M' have the same universes.*
2. *M and M' differ only in the interpretation of fluent symbols.*
3. *There is a ground situation term ξ and a ground action term α such that*
   - *For any ground fluent $F(\chi, \xi)$,*
     $$M'[\![F(\chi,\xi)]\!] = M[\![F(\chi,\xi)]\!].$$
   - $M'[\![do(\alpha,\xi)]\!] = M[\![do(\alpha,\xi)]\!] \neq M[\![\bot]\!]$.
   - *For any ground fluent $F(\chi, do(\alpha,\xi))$, either*
     $$M'[\![F(\chi, do(\alpha,\xi))]\!] = M[\![F(\chi, do(\alpha,\xi))]\!],$$
     or
     $$M'[\![F(\chi, do(\alpha,\xi))]\!] = M[\![F(\chi,\xi)]\!].$$
   - *There exists a ground fluent $F(\chi, do(\alpha,\xi))$ such that*
     $$M'[\![F(\chi, do(\alpha,\xi))]\!] \neq M[\![F(\chi, do(\alpha,\xi))]\!].$$

However, the class of minimally abnormal model of domain (5) does entail neither (6) nor (7). In fact, such a class includes also models in which (7) is not satisfied. As a consequence, in these models, the effects of shooting are canceled.

Definition 2 *Let M be a model of a domain D. We say that actions are maximally executable in M if M satisfies*

$$s = \bot \supset do(a,s) = \bot, \tag{8}$$

and there does not exist a model M' of D satisfying (8) such that

1. *M and M' have the same universes for actions, objects and values. The universe for situations in M is a subset of the universe for situations in M'.*
2. *M and M' differ only in the interpretation of the fluent symbols and the function do.*
3. *There is a ground situation term ξ and a ground action term α such that*
   - *For any ground fluent $F(\chi, \xi)$,*
     $$M'[\![F(\chi,\xi)]\!] = M[\![F(\chi,\xi)]\!].$$
   - $M'[\![do(\alpha,\xi)]\!] \neq M[\![do(\alpha,\xi)]\!] = M[\![\bot]\!]$.

We say that a model M of a domain D is an *intended model* if M is minimally abnormal and actions are maximally executable in M. Of course, each intended model of domain (5) satisfies both (6) and (7).

3 A SECOND ORDER SOLUTION

We assume to have also function variables for fluent symbols. For each fluent symbol F, we denote with F' a fluent variable with the same domain as F. We also assume that $F_1, \ldots, F_n$ are all the fluent symbols in the language and that the domain D consists of a finite number of propositions. By $\bigwedge_D \Upsilon(s)$ [$\bigwedge_D \forall x(\Phi(x,a,s) \supset \Psi(x,a,s'))$, resp.] we mean the conjunction of the constraints [of the right hand side of the effect propositions, resp.] in D. $\phi[t_1/z_1, \ldots, t_m/z_m]$ is the expression obtained via simultaneous substitution of t_i for z_i $(1 \leq i \leq m)$ in ϕ.

We begin with a second order characterization of the class of minimally abnormal models.

$$\begin{array}{l} do(a,s) = s' \wedge s' \neq \bot \supset \neg\exists F_1' \ldots F_n'(\\ \quad \bigwedge_D \Upsilon(s)[s'/s, F_1'/F_1, \ldots, F_n'/F_n] \quad \wedge \\ \quad \bigwedge_D \forall x(\Phi(x,a,s) \supset \Psi(x,a,s')[F_1'/F_1, \ldots, F_n'/F_n]) \wedge \\ \quad \bigwedge_{1 \leq i \leq n} (\forall x_i (F_i'(x_i,s') = F_i(x_i,s') \vee \\ \qquad\qquad F_i'(x_i,s') = F_i(x_i,s))) \quad \wedge \\ \quad \bigvee_{1 \leq i \leq n} (\exists x_i F_i'(x_i,s') \neq F_i(x_i,s'))). \end{array} \tag{9}$$

Theorem 1 *Let M be a model of a domain D. M is minimally abnormal iff M satisfies (9).*

Intuitively, (9) imposes to consider only those models M in which, for each action α and situation ξ such that

$$M \models do(\alpha, \xi) \neq \bot,$$

it is not possible to find fluent symbols $F_1', \ldots, F_n'$

- satisfying the conditions imposed by the constraints and the effect propositions (second and third lines of (9)); and
- for each tuple of objects χ_i,

$$\begin{array}{l} M \models F_i'(\chi_i, do(\alpha,\xi)) = F_i(\chi_i, do(\alpha,\xi)) \vee \\ \qquad F_i'(\chi_i, do(\alpha,\xi)) = F_i(\chi_i, \xi), \end{array}$$

  with $1 \leq i \leq n$ (third and fourth line of (9)); and
- for at least one tuple of objects χ, either

$$M \models F_1'(\chi, do(\alpha,\xi)) \neq F_1(\chi, do(\alpha,\xi)),$$

or

$$M \models F_2'(\chi, do(\alpha,\xi)) \neq F_2(\chi, do(\alpha,\xi)),$$

$\ldots$
or

$$M \models F_n'(\chi, do(\alpha,\xi)) \neq F_n(\chi, do(\alpha,\xi))$$

(last line of (9)).

The class of models in which actions are maximally executable is characterized by the following formula.

$$\begin{array}{l} do(a,s) = s' \supset (s' = \bot \equiv (s = \bot \vee \neg\exists F_1' \ldots F_n'(\\ \quad \bigwedge_D \Upsilon(s)[s'/s, F_1'/F_1, \ldots, F_n'/F_n] \wedge \\ \quad \bigwedge_D \forall x(\Phi(x,a,s) \supset \Psi(x,a,s')[F_1'/F_1, \ldots, F_n'/F_n])))). \end{array} \tag{10}$$

Theorem 2 *Let M be a model of a domain D. Actions are maximally executable in M iff M satisfies (10).*

It can also be proved that a model M of a domain D satisfies (10) if and only if M satisfies (8) and

$$\begin{array}{l} do(a,s) = s' \wedge s' = \bot \supset (s = \bot \vee \neg\exists F_1' \ldots F_n'(\\ \quad \bigwedge_D \Upsilon(s)[s'/s, F_1'/F_1, \ldots, F_n'/F_n] \wedge \\ \quad \bigwedge_D \forall x(\Phi(x,a,s) \supset \Psi(x,a,s')[F_1'/F_1, \ldots, F_n'/F_n]))). \end{array}$$

4 REDUCTION TO FIRST ORDER

In general, it is not possible to reduce (9) to an equivalent first order formula. The following example is from [Lin and Reiter, 1994].

Example 1 *Let P and R be ternary fluent symbols and A an action constant. Consider the propositional domain consisting of the effect proposition*

$$do(a,s) = s' \wedge s' \neq \bot \supset (a = A \wedge P(x,y,s) \supset R(x,y,s')),$$

and of the constraints

$$\begin{array}{l} R(x,x,s), \\ R(x,y,s) \supset R(y,x,s), \\ R(x,y,s) \wedge R(y,z,s) \supset R(x,z,s). \end{array}$$

It is not possible to characterize the set of minimally abnormal models of D with a first order theory. Otherwise, such a theory, together with the first order formula

$$do(A, S_0) \neq \bot \wedge \neg R(x,y,S_0)$$

would entail the transitive closure of $P(x,y,S_0)$ (which is not first order definable).

However, there are some special cases in which both (9) and (10) have first order equivalent formulas.

We say that a formula is *solitary in* $F(x,s')$[1] if it has the form

$$\exists y \forall x \phi(y,x,a,s,s')$$

where

[1] This terminology is suggested by a vague analogy with the corresponding concept in [Lifschitz, 1985].

- y and x are tuples (eventually empty) of object variables, and
- for each occurrence in $\phi(y, x, a, s, s')$ of a fluent $G(\chi, \xi)$, if ξ is s' then $G(\chi, \xi)$ is $F(x, s')$.

For example, the first, second and third propositions in (5) are solitary in $Loaded(s')$, $Alive(s')$ and $Loaded(s')$ respectively.

We say that a formula is *separable in* s' if it is a combination of conjunctions and disjunctions of solitary formulas in some fluent $F(x, s')$.

The formula $(Walking(s') = \mathsf{F} \vee Alive(s') = \mathsf{T})$ is separable in s'.

Theorem 3 *Let D be a domain such that for each constraint (1) and effect proposition (2), $\Upsilon(s)$ and $\forall x(\Phi(x, a, s) \supset \Psi(x, a, s'))$ are separable in s and s' respectively. Then both (9) and (10) have a first order equivalent formula.*

Consider e.g. (9). In the hypotheses of Theorem 3, we can rewrite

$$\begin{array}{l}\exists F_1' \ldots F_n'(\\ \quad \bigwedge_D \Upsilon(s)[s'/s, F_1'/F_1, \ldots, F_n'/F_n] \quad \wedge \\ \quad \bigwedge_D \forall x(\Phi(x, a, s) \supset \Psi(x, a, s')[F_1'/F_1, \ldots, F_n'/F_n]) \wedge \\ \quad \bigwedge_{1 \leq i \leq n} (\forall x_i (F_i'(x_i, s') = F_i(x_i, s') \vee \\ \qquad\qquad F_i'(x_i, s') = F_i(x_i, s))) \quad \wedge \\ \quad \bigvee_{1 \leq i \leq n} (\exists x_i F_i'(x_i, s') \neq F_i(x_i, s'))).\end{array}$$

in a form

$$\exists F_1' \ldots F_n' \exists y \bigvee_i \bigwedge_{1 \leq j \leq n} \forall x_j \phi_{ij}(y, x_j, a, s, s'), \qquad (11)$$

by first order manipulation. Furthermore, we can assume that each occurrence in $\phi_{ij}(y, x_j, a, s, s')$ of a fluent with a fluent variable has the form $F_j'(x_j, s')$.

Then, (11) can be rewritten as

$$\exists y \bigvee_i \bigwedge_{1 \leq j \leq n} \exists F_j' \forall x_j s''(s'' = s' \supset \phi_{ij}(y, x_j, a, s, s')[s''/s'])$$

which can be reduced to a first order formula using the equivalence (see e.g. [Enderton, 1972])

$$\models \forall x s \exists v \phi(x, v, s) \equiv \exists F \forall x s \phi(x, v, s)[F(x, s)/v].$$

There are two important classes of domains meeting the conditions of Theorem 3 and thus for which we can find first order formulas equivalent to (9) and (10).

To the first class belong the propositional domains in which the constraints are without situation terms and the effect propositions have one of the following two forms

$$do(a, s) = s' \wedge s' \neq \bot \supset \forall x(P(x, a, s) \supset F(x, s')),$$
$$do(a, s) = s' \wedge s' \neq \bot \supset \forall x(Q(a, x, s) \supset \neg F(x, s')).$$

For example, the propositional domain (5) without the constraint $Walking(s) \supset Alive(s)$, belongs to this set. These are essentially the domains for which Reiter [Reiter, 1991] proposes a method for the generation of "successor state axioms".

To the second class belong the domains in which each proposition does not contain object terms. For example, the domain (5) belongs to this set. For such domains, (9) is equivalent to

$$\begin{array}{l}do(a, s) = s' \wedge s' \neq \bot \supset \neg \exists v_1 \ldots v_n(\\ \quad \bigwedge_D \Upsilon(s)[v_1/F_1(s), \ldots, v_n/F_n(s)] \quad \wedge \\ \quad \bigwedge_D (\Phi(a, s) \supset \Psi(a, s')[v_1/F_1(s'), \ldots, v_n/F_n(s')]) \wedge \\ \quad \bigwedge_{1 \leq i \leq n} (v_i = F_i(s') \vee v_i = F_i(s)) \quad \wedge \\ \quad \bigvee_{1 \leq i \leq n} (v_i \neq F_i(s'))),\end{array} \qquad (12)$$

while (13) is equivalent to:

$$\begin{array}{l}do(a, s) = s' \supset (s' = \bot \equiv (s = \bot \vee \neg \exists v_1 \ldots v_n(\\ \quad \bigwedge_D \Upsilon(s)[v_1/F_1(s), \ldots, v_n/F_n(s)] \wedge \\ \quad \bigwedge_D (\Phi(a, s) \supset \Psi(a, s')[v_1/F_1(s'), \ldots, v_n/F_n(s')])))).\end{array} \qquad (13)$$

For example, in the case of domain (5), (12) and (13) become[2]

$$do(a, s) = s' \wedge s' \neq \bot \supset Res(a, Loaded(s'), Alive(s'), Alive(s'), s)$$

and

$$do(a, s) = \bot \equiv s = \bot \vee \neg \exists v_1 v_2 v_3 Res^0(a, v_1, v_2, v_3, s)$$

respectively.

The predicate $Res^0(a, v_1, v_2, v_3, s)$ is defined by the formula

$$\begin{array}{l}(a = Load \supset v_1 = \mathsf{T}) \quad \wedge \\ (a = Shoot \wedge Loaded(s) = \mathsf{T} \supset v_2 = \mathsf{F}) \wedge \\ (a = Shoot \supset v_1 = \mathsf{F}) \quad \wedge \\ (v_3 = \mathsf{T} \supset v_2 = \mathsf{T}),\end{array}$$

while $Res(a, v_1, v_2, v_3, s)$ is defined by:

$$\begin{array}{l}\neg \exists w_1 w_2 w_3 (Res^0(a, w_1, w_2, w_3, s) \quad \wedge \\ \qquad (w_1 = v_1 \vee w_1 = Loaded(s)) \quad \wedge \\ \qquad (w_2 = v_2 \vee w_2 = Alive(s)) \quad \wedge \\ \qquad (w_3 = v_3 \vee w_3 = Walking(s)) \quad \wedge \\ \qquad (w_1 \neq v_1 \vee w_2 \neq v_2 \vee w_3 \neq v_3)).\end{array}$$

[2] Notice that we do not need to consider the constraints in the last line of (5) since without occurrences of fluents.

The following example is Example 3 in [Kartha and Lifschitz, 1995].

Example 2 *Consider a table divided into three locations F, G and H. A block can be in exactly one of these three locations. There is an action A, which when performed when the block is in location F moves it out of that location. Hence, after the action is performed, the block is in location G or H, but we do not know which.*

This scenario is formalized by the following propositional domain:

$$do(a, s) = s' \wedge s' \neq \bot \supset (a = A \wedge F(s) \supset \neg F(s')),$$
$$F(s) \vee G(s) \vee H(s),$$
$$\neg F(s) \vee \neg G(s),$$
$$\neg F(s) \vee \neg H(s),$$
$$\neg G(s) \vee \neg H(s).$$

This domain admits a first order reduction a-priori computable. Furthermore, because of the ternary constraint, the methods developed in [Pinto, 1995] for the generation of successor state axioms are not applicable.

5 RELATIONS WITH $\mathcal{AR}$

$\mathcal{AR}$ [Giunchiglia *et al.*, 1996] is an "(high level) action description language" in the spirit of $\mathcal{A}$ from [Gelfond and Lifschitz, 1992]. Preliminary reports on $\mathcal{AR}$ appeared in [Kartha and Lifschitz, 1994] and [Giunchiglia *et al.*, 1995]. Differently from these two previous reports, [Giunchiglia *et al.*, 1996] incorporates the idea from [Lifschitz, 1996] of distinguishing between the "action description" and "action query" components of an action language. Intuitively, an action description language is for describing the scenario, i.e. its possible states and the effects of executing actions in states. An action query language is for asking queries about the scenario. A typical query is whether a certain condition is true after executing a given sequence of actions. The sum of an action description and an action query language defines an action language.

In this section we first review the syntax (Section 5.1) and semantics (Section 5.2) of $\mathcal{AR}$. In Section 5.3 we introduce a query language that can be used, together with $\mathcal{AR}$, to address temporal projection problems. These first three subsections are directly derived from [Giunchiglia *et al.*, 1996]. In Section 5.4, we provide a correct and complete translation of any domain description $\mathcal{D}$ of $\mathcal{AR}$ in a domain D_F in the situation calculus. Finally, in Section 5.6, we show that the models of D_F are exactly the intended models of the straightforward translation of $\mathcal{D}$ in the situation calculus.

5.1 SYNTAX OF $\mathcal{AR}$

$\mathcal{AR}$ is a family of "action description languages". An $\mathcal{AR}$ language is characterized by

- a nonempty set of fluent symbols (or fluent names),
- a function, associating with every fluent name F a nonempty set Rng_F of symbols that is called the *range* of F,
- a subset of fluent names, which are said to be *inertial*,
- a nonempty set of symbols, that are called *action names*, or *actions*.

A *value* is a symbol that belongs to the range of a fluent.

An *atomic formula* has the form (F **is** V), where F is a fluent symbol, and $V \in Rng_F$. A *formula* is a propositional combination of atomic formulas. In the following, *True* stands for some fixed tautology and *False* for $\neg$*True*. We also say that a fluent F is *propositional* if $Rng_F = \{\mathsf{F}, \mathsf{T}\}$. For a propositional fluent F, we abbreviate the atomic formula

$$F \textbf{ is } \mathsf{T}$$

by F.

There are three types of *propositions* in an $\mathcal{AR}$ language: determinate effect propositions, indeterminate effect propositions, and constraints. A *determinate effect proposition* is an expression of the form

$$A \textbf{ causes } C \textbf{ if } P, \tag{14}$$

where A is an action, and C and P are formulas. An *indeterminate effect proposition* is an expression of the form

$$A \textbf{ possibly changes } F \textbf{ if } P, \tag{15}$$

where A is an action, F an inertial fluent symbol and P a formula. In effect propositions (14), (15), the part

$$\textbf{if } P$$

is dropped if P is *True*.

Finally, if C is a formula then a *constraint* has the form

$$\textbf{always } C. \tag{16}$$

An *action description* is a set of propositions.

For example, the "murder mystery" can be formalized in $\mathcal{AR}$ as follows (all the fluents are propositional and inertial):

$$\begin{array}{l} \textit{Load} \textbf{ causes } \textit{Loaded}, \\ \textit{Shoot} \textbf{ causes } \neg\textit{Alive} \textbf{ if } \textit{Loaded}, \\ \textit{Shoot} \textbf{ causes } \neg\textit{Loaded}, \\ \textbf{always } \textit{Walking} \supset \textit{Alive}. \end{array} \tag{17}$$

5.2 SEMANTICS OF $\mathcal{AR}$

The semantics of an action description is defined in terms of a transition system. A transition system is characterized by a set of states, and a transition function mapping a state σ and an action A into a set of states —the states resulting from the execution of A in σ.

A *valuation* is a function that associates to each fluent symbol F an element of Rng_F. A valuation σ is extended to atomic formulas as follows

$$\sigma(F \textbf{ is } V) = \begin{cases} \mathsf{T}, \text{ if } \sigma(F) = V, \\ \mathsf{F}, \text{ otherwise.} \end{cases}$$

and to arbitrary formulas according to the truth tables of propositional logic.

Consider an action description $\mathcal{A}$. A valuation σ is said to be a *state* if, for every constraint (16) in $\mathcal{A}$, $\sigma(C) = \mathsf{T}$. The action description (17) has six states, corresponding to the six possible valuations satisfying the constraint $\textit{Walking} \supset \textit{Alive}$.

The transition function *Res* associated to each action description is defined in two steps.

First, $Res^0(A, \sigma)$ is defined to be the set of states σ' such that, for each determinate effect proposition (14) in $\mathcal{A}$, $\sigma'(C) = \mathsf{T}$ whenever $\sigma(P) = \mathsf{T}$. Then, $Res(A, \sigma)$ is defined to be the set of states σ' such that for any state σ'' in $Res^0(A, \sigma)$, $New_A(\sigma, \sigma'')$ is not a proper subset of $New_A(\sigma, \sigma')$. $New_A(\sigma, \sigma')$ is the set of formulas

$$F \textbf{ is } \sigma'(F)$$

such that

- F is inertial and $\sigma'(F) \neq \sigma(F)$, or
- for some indeterminate effect proposition (15) in $\mathcal{A}$, $\sigma(P) = \mathsf{T}$.

5.3 TEMPORAL PROJECTION IN $\mathcal{AR}$

The specification of a temporal projection problem includes an action description and a set of "initial conditions" that have to be satisfied by the initial situation. An initial condition is specified by a proposition of the form

$$\textbf{initially } C \tag{18}$$

where C is a formula in the syntax of $\mathcal{AR}$. We will use the term *domain description* to mean a set of propositions consisting of an action description union a set of initial conditions. (Thus an action description is also a domain description.)

Given a domain description, a temporal projection problem consists in the check of whether a certain condition is true after executing a given sequence of actions starting from the initial situation. To specify such a condition we introduce *value propositions* as expressions of the form

$$C \textbf{ after } A_1; \ldots; A_n \tag{19}$$

where C is a formula and $A_1, \ldots, A_n$ ($n > 0$) are action names.

A *history* for an action description $\mathcal{A}$ is a finite sequence

$$\sigma_0, A_1, \sigma_1, \ldots, A_n, \sigma_n \tag{20}$$

($n \geq 0$) such that σ_0 is a state and

$$\sigma_i \in Res(A_i, \sigma_{i-1}) \qquad (1 \leq i \leq n).$$

A history (20) *satisfies* an initial condition (18) if σ_0 satisfies C. Given an action description $\mathcal{A}$ and a set $\mathcal{I}$ of initial conditions, we say that (19) is a *consequence* of the domain description $\mathcal{A} \cup \mathcal{I}$ if, for any history (20) satisfying all the initial conditions in $\mathcal{I}$, $\sigma_n(C) = \mathsf{T}$.

For example, the domain description (17) entails

$$\neg\textit{Alive} \textbf{ after } \textit{Load}; \textit{Wait}; \textit{Shoot}.$$

If we add the initial condition

$$\textbf{initially } \textit{Loaded} \tag{21}$$

then, (17) together with (21) entails also

$$\neg\textit{Alive} \textbf{ after } \textit{Wait}; \textit{Shoot}.$$

5.4 TRANSLATION OF $\mathcal{AR}$ INTO FIRST ORDER LOGIC

We restrict our attention to *finite* domain descriptions, that is to domain descriptions with finitely many fluent symbols, values, actions and propositions. We also assume that $F_1, \ldots, F_m$ [$F_{m+1}, \ldots, F_n$, resp.] are the inertial [not inertial, resp.] fluent symbols, listed according to a fixed enumeration.

We will translate each finite domain description $\mathcal{D}$ into a many sorted first order theory D_F in the language L_F.

5.4.1 Definition of the language L_F

As in Section 2, the language L_F will be many sorted. We have sorts for situations (*situation*), actions (*action*) and values (*value*). We adopt the same conventions used in Section 2.

For the constants, we assume that all the values of $\mathcal{D}$, all actions of $\mathcal{D}$, and the situation constants S_0 and $\perp$ are declared as constants of the appropriate sort. The function symbols are:

- for each fluent symbol F_i, a unary function F_i: *situation* $\rightarrow$ *value*,
- the function *do*: *action* $\times$ *situation* $\rightarrow$ *situation*.

Finally, the predicate constants are:

- $State_F$ whose arguments are an n-tuple of values,
- Res_F^0 whose arguments are an action, an n-tuple of values, and a situation,
- Res_F whose arguments are an action, an n-tuple of values, and a situation.

Each predicate is meant to capture the corresponding concept of $\mathcal{AR}$ semantics.

5.4.2 Unique Name Assumptions

For each two distinct action names A_i and A_j,

$$A_i \neq A_j. \tag{22}$$

For each fluent symbol F_i and for each pair V_j, V_k of distinct values in $Rng(F_i)$,

$$V_j \neq V_k. \tag{23}$$

Finally,

$$S_0 \neq \perp. \tag{24}$$

5.4.3 Definition of $State_F$, Res_F^0 and Res_F

For any formula C of $\mathcal{D}$

- with C_v we indicate the formula of L_F obtained replacing each fluent symbol F_i with the variable v_i in C,
- for any situation term ξ, with C_ξ we indicate the formula of L_F obtained replacing each fluent symbol F_i with the term $F_i(\xi)$ in C.

The next three equations define the predicates $State_F$, Res_F^0 and Res_F, respectively. $State_F(v_1, \ldots, v_n)$ is defined to be equivalent to:

$$\bigwedge_{1 \leq i \leq n} \Big(\bigvee_{V \in Rng(F_i)} v_i = V \Big) \wedge \bigwedge_{\textbf{always } C \in \mathcal{D}} C_v. \tag{25}$$

Proposition 1 *Let M be a structure of L_F satisfying* (23) *and* (25). *A set $\{\langle F_i, V_i\rangle \mid 1 \leq i \leq n\}$ is state of $\mathcal{D}$ iff $M \models State_F(V_1, \ldots, V_n)$.*

Proposition 1 trivially follows from the fact that in a model equality is interpreted as identity.

$Res_F^0(a, v_1, \ldots, v_n, s)$ is defined equivalent to:

$$\begin{array}{c} State_F(v_1, \ldots, v_n) \wedge \\ \bigwedge_A \textbf{ causes } C \textbf{ if } _{P \in \mathcal{D}}((a = A \wedge P_s) \supset C_v). \end{array} \tag{26}$$

Given a structure M of L_F, we say that a ground situation term ξ *situates* a valuation σ of $\mathcal{D}$ if

$$M[F_i(\xi)] = M[\sigma(F_i)] \qquad (1 \leq i \leq n).$$

If ξ is a ground situation term situating σ in M then, for any formula P of $\mathcal{D}$, $\sigma(P) = \mathsf{T}$ iff $M \models P_\xi$, (this can be proved by induction on the structure of the formula).

Proposition 2 *Let A be an action and σ a valuation of $\mathcal{D}$. Let M be a structure of L_F satisfying* (22)–(26) *and ξ a ground situation term situating σ in M. $\sigma' \in Res^0(A, \sigma)$ iff $M \models Res_F^0(A, \sigma'(F_1), \ldots, \sigma'(F_n), \xi)$.*

The thesis follows from Proposition 1 and the uniqueness of names both for actions and values within each range associated to a fluent.

$Res_F(a, v_1, \ldots, v_n, s)$ is defined equivalent to:

$$\begin{array}{l} Res_F^0(a, v_1, \ldots, v_n, s) \wedge \\ \neg \exists w_1 \ldots w_n (Res_F^0(a, w_1, \ldots, w_n, s) \quad \wedge \\ \qquad \bigwedge_{1 \leq i \leq m}(w_i = v_i \vee w_i = F_i(s)) \quad \wedge \\ \qquad \bigvee_{1 \leq i \leq m}(w_i \neq v_i) \quad \wedge \\ \qquad \bigwedge \quad ((a = A \wedge P_s) \supset v_i = w_i)). \\ \qquad A \textbf{ possibly changes } F_i \textbf{ if } P \in \mathcal{D} \end{array} \tag{27}$$

Proposition 3 *Let A be an action and σ a valuation of $\mathcal{D}$. Let M be a structure of L_F satisfying* (22)–(27) *and ξ a ground situation term situating σ in M. $\sigma' \in Res(A, \sigma)$ iff $M \models Res_F(A, \sigma'(F_1), \ldots, \sigma'(F_n), \xi)$.*

The proof is based on the following fact about $\mathcal{AR}$. $\sigma' \in Res(A, \sigma)$ iff $\sigma' \in Res^0(A, \sigma)$ and there is not a valuation $\sigma'' \in Res^0(A, \sigma)$ such that:

- $\sigma''(F) = \sigma'(F)$ for each fluent symbol F such that for some indeterminate effect proposition (15) in $\mathcal{D}$, $\sigma(P) = \mathsf{T}$;
- for each inertial fluent symbol F, either $\sigma''(F) = \sigma'(F)$ or $\sigma''(F) = \sigma(F)$; and
- there exists at least one inertial fluent symbol F such that $\sigma''(F) \neq \sigma'(F)$.

5.4.4 Model conditions

Each initial condition has to be satisfied:

$$\bigwedge_{\textbf{initially } C \in \mathcal{D}} C_{S_0}. \tag{28}$$

Also the constraints have to be satisfied:

$$State_F(F_1(s), \ldots, F_n(s)). \tag{29}$$

If an action A is executable in a state σ, the resulting states shall belong to $Res(A, \sigma)$:

$$do(a, s) \neq \bot \supset Res_F(a, F_1(do(a, s)), \ldots, F_n(do(a, s)), s). \tag{30}$$

Normally, actions are executable:

$$do(a, s) = \bot \equiv s = \bot \vee \neg\exists v_1 \ldots v_n Res^0_F(a, v_1, \ldots, v_n, s). \tag{31}$$

5.5 CORRECTNESS AND COMPLETENESS OF THE TRANSLATION

Let $\mathcal{D}$ be a finite domain description. The *translation of $\mathcal{D}$ in first order logic* is the first order theory D_F consisting of (22)–(31).

For any string of actions $A_1 \ldots A_n$ ($n \geq 0$) we abbreviate the term $do(A_n, do(A_{n-1}, \ldots, do(A_1, S_0)\ldots))$ with $do(A_1 \ldots A_n, S_0)$.

Theorem 4 *Let D_F be the translation of a finite domain description $\mathcal{D}$ in first order logic. A value proposition C* **after** *$\overline{A}$ is a consequence of $\mathcal{D}$ iff D_F entails $do(\overline{A}, S_0) \neq \bot \supset C_{do(\overline{A}, S_0)}$.*

Notice that, domains

- having at least one model in the action language $\mathcal{A}$ [Gelfond and Lifschitz, 1992], or
- without "releases" propositions in the action language $\mathcal{AR}_0$ [Kartha and Lifschitz, 1994], or
- without dependent fluents in the action language $\mathcal{ARD}$ [Giunchiglia and Lifschitz, 1995],

can be regarded as $\mathcal{AR}$ domains. Hence, the above translation and results can be applied also to them.

5.6 MODELS OF THE TRANSLATION

Consider a finite domain description $\mathcal{D}$. Let D_F the translation of $\mathcal{D}$ in first order logic. Define D to be the domain (in the language L_F of D_F, see Section 5.4.1) consisting of:

- the formulas (22)–(24); and
- the effect propositions

  $$do(A, s) = s' \wedge s' \neq \bot \supset (P_s \supset C_{s'})$$

  for each proposition (14) in $\mathcal{D}$; and
- the constraints

  $$C_s$$

  for each proposition (16) in $\mathcal{D}$, and

  $$\bigvee_{V \in Rng(F_i)} F_i(s) = V$$

  for each fluent symbol F_i.

Theorem 5 *Let $\mathcal{D}$ be a finite domain description such that*

- *all the fluent symbols are inertial, and*
- *$\mathcal{D}$ consists only of determinate effect propositions and constraints.*

A structure M of L_F is a model of D_F iff M is an intended model of D.

To check the above theorem, it is enough to

1. unfold the definitions of $State_F$, Res^0_F and Res_F in (29), (30) and (31), and
2. compare the result with D union (12) and (13) (we remind that these two last formulas characterize the set of intended models for domains without object terms): the two theories are logically equivalent.

As a trivial consequence of Theorem 4 and Theorem 5 we have the following Corollary.

Corollary 1 *In the hypotheses of Theorem 5, a value proposition C* **after** *$\overline{A}$ is a consequence of $\mathcal{D}$ iff the class of intended models of D entails $do(\overline{A}, S_0) \neq \bot \supset C_{do(\overline{A},S_0)}$.*

Acknowledgments

This work has been initiated while the author was joining the "Logical Foundations of Artificial Intelligence Group" in the Computer Science Department of the University of Texas at Austin. Thanks to Neelakantan Kartha and Norm McCain for useful discussions on subjects related to this paper. Thanks to Vladimir Lifschitz and Hudson Turner for very useful comments on a early version of this paper.

This work has been supported by the Italian National Research Council (CNR), under the projects "Sistemi Autonomi Ragionanti e Intelligenti (SARI)" and "Analisi Sperimentale delle proprietà computazionali delle logiche per la rappresentazione della conoscenza".

References

[Baker, 1991] Andrew Baker. Nonmonotonic reasoning in the framework of situation calculus. *Artificial Intelligence*, 49:5–23, 1991.

[Doherty *et al.*,] Patrick Doherty, Witold Lukaszewicz, and Andrzey Szalas. Computing circumscription revisited: A reduction algorithm. In *Proc. IJCAI-95*, pages 1502–1508.

[ECAI, 1996] ECAI. Workshop on Integrating Nonmonotonicity into Automated Reasoning Systems, 1996.

[Enderton, 1972] H.B. Enderton. *A Mathematical Introduction to Logic.* Academic Press, 1972.

[Gabbay and Ohlbach, 1992] Dov Gabbay and Hans Ohlbach. Quantifier elimination in second-order predicate logic. In Bernhard Nebel, Charles Rich, and William Swartout, editors, *Proc. of the Third Int'l Conf. on Principles of Knowledge Representation and Reasoning*, pages 425–435, 1992.

[Gelfond and Lifschitz, 1992] Michael Gelfond and Vladimir Lifschitz. Representing actions in extended logic programming. In Krzysztof Apt, editor, *Proc. Joint Int'l Conf. and Symp. on Logic Programming*, pages 559–573, 1992.

[Giunchiglia and Lifschitz, 1995] Enrico Giunchiglia and Vladimir Lifschitz. Dependent fluents. In *Proc. IJCAI-95*, 1995.

[Giunchiglia *et al.*, 1995] Enrico Giunchiglia, G. Neelakantan Kartha, and Vladimir Lifschitz. Actions with indirect effects (extended abstract). In *Working Notes of the AAAI Spring Symposium 95 on Extending Theories of Actions*, 1995.

[Giunchiglia *et al.*, 1996] Enrico Giunchiglia, G. Neelakantan Kartha, and Vladimir Lifschitz. Actions with indirect effects. Manuscript. Extended and revised version of [Giunchiglia *et al.*, 1995]., 1996.

[IJCAI, 1995] IJCAI. Workshop on Applications and Implementations of Nonmonotonic Reasoning Systems, 1995.

[Kartha and Lifschitz, 1994] G. Neelakantan Kartha and Vladimir Lifschitz. Actions with indirect effects (preliminary report). In *Proc. of the Fourth Int'l Conf. on Principles of Knowledge Representation and Reasoning*, pages 341–350, 1994.

[Kartha and Lifschitz, 1995] G. Neelakantan Kartha and Vladimir Lifschitz. A simple formalization of actions using circumscription. In *Proc. IJCAI-95*, 1995.

[Kartha, 1993] G. Neelakantan Kartha. Soundness and completeness theorems for three formalizations of action. In *Proc. of IJCAI-93*, pages 724–729, 1993.

[Kartha, 1994] G. Neelakantan Kartha. Two counterexamples related to Baker's approach to the frame problem. *Artificial Intelligence*, 69:379–391, 1994.

[Kartha, 1996] G. Neelakantan Kartha. On the range of applicability of Baker's approach to the frame problem. In *Proc. of AAAI-96*, 1996.

[Lifschitz, 1985] Vladimir Lifschitz. Computing circumscription. In *Proc. of IJCAI-85*, pages 121–127, 1985.

[Lifschitz, 1990] Vladimir Lifschitz. Frames in the space of situations. *Artificial Intelligence*, 46:365–376, 1990.

[Lifschitz, 1995] Vladimir Lifschitz. Nested abnormality theories. *Artificial Intelligence*, 74:351–365, 1995.

[Lifschitz, 1996] Vladimir Lifschitz. Two components of an action language. In *Working Papers of*

the Third Symposium on Logical Formalizations of Commonsense Reasoning, 1996.

[Lin and Reiter, 1994] Fangzhen Lin and Raymond Reiter. State constraints revisited. *Journal of Logic and Computation*, 4:655–678, 1994.

[Lin and Shoham, 1991] Fangzhen Lin and Yoav Shoham. Provably correct theories of action (preliminary report). In *Proc. AAAI-91*, pages 349–354, 1991.

[Pednault, 1989] Edwin Pednault. ADL: Exploring the middle ground between STRIPS and the situation calculus. In Ronald Brachman, Hector Levesque, and Raymond Reiter, editors, *Proc. of the First Int'l Conf. on Principles of Knowledge Representation and Reasoning*, pages 324–332, 1989.

[Pinto, 1995] Andres Javier Pinto. *Temporal Reasoning in the Situation Calculus.* PhD thesis, 1995. PhD thesis, University of Toronto.

[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.

[Winslett, 1988] Marianne Winslett. Reasoning about action using a possible model approach. In *Proc. AAAI-88*, pages 89–93, 1988.

Embracing Occlusion in Specifying the Indirect Effects of Actions

Joakim Gustafsson
Department of Computer
and Information Science
Linköping University
S-58183 Linköping, Sweden
joagu@ida.liu.se

Patrick Doherty
Department of Computer
and Information Science
Linköping University
S-58183 Linköping, Sweden
patdo@ida.liu.se

Abstract

In this paper, we extend PMON, a logic for reasoning about action and change, with causal rules which are used to specify the indirect effects of actions. The extension, called PMON(RCs), has the advantage of using explicit time, includes actions with durations, nondeterministic actions, allows partial specification of the timing and order of actions and has been assessed correct for at least the $\mathcal{K}$-IA class of action scenarios within the Features and Fluents framework. Most importantly, the circumscription policy used is easily shown to be reducible to the first-order case which insures that standard theorem proving techniques and their optimizations may be used to compute entailment. In addition, we show how the *occlusion* concept previously used to deal with duration and nondeterministic actions proves to be equally versatile in representing causal constraints and delayed effects of actions. We also discuss related work and consider the strong correspondence between our work and recent work by Lin, who uses a *Cause* predicate to specify indirect effects similar to our use of *Occlude* in PMON, and a minimization policy related to that used in PMON.

1 Introduction

Sandewall [17] has recently proposed a systematic approach to the representation of knowledge about dynamical systems that includes a framework in which to assess the range and applicability of existing and new logics of action and change. As part of the framework, several logics of action and change are introduced and assessed correct for particular classes of action scenario descriptions. The most general class $\mathcal{K} - IA$ and one of it's associated entailment relations, PMON, permits scenarios with nondeterministic actions, actions with duration, partial specifications at any state in the scenario, context dependency, and incomplete specification of the timing and order of actions. Doherty [2, 1] provides a syntactic characterization of PMON in terms of circumscription and classical logic and shows that for the $\mathcal{K} - IA$ class, the circumscription axiom can be reduced to a 1st-order formula. Although PMON is assessed correct for a broad class of action scenarios, it is restricted to actions that do not permit indirect effects. It deals with the simple frame problem and not ramification.

Inspired by the use of the frame construct in Kartha and Lifschitz [8], Doherty and Peppas [4] have extended PMON with the frame construct to deal with various types of ramification. Lifschitz and Kartha capitalize on the *frame* concept discussed in Lifschitz [10] which is used in this case to specify causal dependencies between direct and indirect effects of actions. In addition, they use both a *release* construct and a *filtering* method, analogous to the *occlusion* construct and filtering method first proposed by Sandewall [16, 17] for dealing with postdiction, actions with duration and nondeterminism. The novelty of Kartha and Lifschitz's approach is a tripartite division of fluents into frame, frame released, and non-frame fluents which is used to deal with certain types of causally directed ramification. In Doherty and Peppas[4], we considered the relation between PMON(R) and $\mathcal{AR}_0$. We show that PMON(R) subsumes $\mathcal{AR}_0$ in several different respects, while $\mathcal{AR}_0$ has more expressibility in other respects. The extension, PMON(R), is also characterized in terms of a circumscription axiom, but in contrast to the case for PMON, it can not in the general case be reduced to a first-order formula.

Recently, causal minimization techniques have again become increasingly more popular. These techniques

are based on introducing explicit causal predicates or causal rules to specify the indirect effects of actions([11], [12], [15], [20]). In this paper, we will show how the base logic PMON can be extended with *causal rules* with little change to the existing formalism. In fact, causal rules in this context are really nothing more than macros in a surface language which when translated into the base language of PMON take advantage of the already existing predicate *Occlude*. As stated previously,the *Occlude* predicate has already been proven to be quite versatile in specifying actions with duration and indeterminate effects of actions. One of the benefits of using this approach for specifying indirect effects of actions is that the original circumscription policy for the base logic PMON is virtually left intact. Consequently, the extended version which we will call PMON(RCs), inherits the nice feature of allowing the reduction of any circumscribed action theory to a logically equivalent first-order theory provided the axiomatization of the chosen flow of time is first-order definable.

In Doherty and Peppas [4], we showed how the *release* predicate used by Lifschitz and Kartha had the same function as *Occlude* in regard to simple forms of nondeterministic actions. What is even more striking when investigating PMON(RCs) is the comparison with Lin's recent proposals for dealing with indirect effects of actions and indeterminate actions ([11, 12]). In fact, the *Cause* predicate and minimization policy used by Lin are virtually analogous with the use of the *Occlude* predicate and the minimization policy used in both the original PMON [17, 2], and the minor extension made in PMON(RCs).

There are a number of factors which make a formal comparison between the two approaches difficult. In particular, we use a linear time structure, whereas Lin uses the situation calculus, although we should mention that we initiated a study of the proper formal tools needed to do such comparisons in Doherty and Peppas[4]. In addition, Lin also deals with actions which may fail and the qualification problem. We have not yet extended our formalism to deal with these issues. Finally, much of Lin's work deals with the automatic generation of successor state axioms as a means for computing entailment in the situation calculus framework. Although we are currently working on implementations of PMON and its various extensions, formally we have only gone as far as providing an algorithm for automatic reduction of any circumscribed action theory within our framework to the logically equivalent first-order case.

In the rest of the paper, we will do the following: (1) Briefly introduce the base version of PMON. (2) Extend it with causal and acausal constraints resulting in PMON(RCs). (3) Show that any circumscribed action scenario in PMON(RCs) is reducible to the first-order case. (4) Compare PMON(RCs) with a number of recent proposals in the literature, in particular Lin's, Thielscher's and Sandewall's approaches. (5) Conclude with a discussion.

2 Action Scenario Descriptions

Many reasoning problems involving action and change can be conveniently represented in terms of *(action)scenario descriptions* . Scenario descriptions can be described as partial specifications of an initial and other states of a system, combined with descriptions of some of the actions that have occurred together with their timing. The "Yale Shooting" or "Stanford Murder Mystery" problems are well known examples of scenario descriptions. Scenario descriptions can be described directly in terms of a logical language, or for convenience, described first in a higher level macro language which is then compiled into a logical language. In our framework, we will represent action scenarios in a surface language $\mathcal{L}(\mathcal{SD})$, which will then be translated into a standard logical language $\mathcal{L}(\mathcal{FL})$. All formal reasoning will be done using $\mathcal{L}(\mathcal{FL})$ together with appropriate circumscription policies for modeling various inertia policies. Detailed descriptions of both languages and the translation process may be found in [1, 2]. In the following sections, we will provide sufficient detail to follow the examples in this paper.

2.1 The Language $\mathcal{L}(\mathcal{SD})$

The formal syntax for specifying scenario descriptions is defined in terms of the surface language $\mathcal{L}(\mathcal{SD})$, consisting of action occurrence statements, action law schemas, and observation statements, labeled with the symbols "ac", "acs", and "obs", respectively. The well known Stanford Murder Mystery scenario is shown below using the $\mathcal{L}(\mathcal{SD})$ syntax:[1]

Example 1

$obs0$	$\mathtt{t}_0 = 0 \wedge \mathtt{t}_1 = 10$
$obs1$	$[\mathtt{t}_0]alive$
$obs2$	$[\mathtt{t}_1]\neg alive$
$ac1$	$[2,6]Fire$

[1]Note that $obs0$ is not strictly necessary. It is used here simply to show that observation statements may contain arbitrary temporal constraints associated with a particular scenario.

$$acs1 \quad [t_1, t_2]Fire \rightsquigarrow ([t_1]loaded \rightarrow [t_1, t_2](alive := F \wedge loaded := F)$$

Given a scenario description Υ, consisting of statements in the surface language $\mathcal{L}(\mathcal{SD})$, these statements are translated into formulas in the many sorted first-order language $\mathcal{L}(\mathcal{FL})$ via a two-step process. In the first step, action schemas in Υ are instantiated with each action occurrence statement of the same name, resulting in what are called *schedule statements* (Each schedule statement is labeled with the symbol "scd"). The resulting schedule statements replace the action schemas and action occurrence statements. The result of the first step is an *expanded (action) scenario description*, Υ', consisting of both schedule and observation statements. The expanded scenario description associated with Example 1 is shown below:

Example 2

$$obs0 \quad t_0 = 0 \wedge t_1 = 10$$
$$obs1 \quad [t_0]alive$$
$$obs2 \quad [t_1]\neg alive$$
$$scd1 \quad ([2]loaded \rightarrow [2, 6](alive := F \wedge loaded := F)$$

In the second step of the translation process, macro-translation definitions are used to translate statements in Υ' into formulas in $\mathcal{L}(\mathcal{FL})$. Before applying this step to the example, we must first define $\mathcal{L}(\mathcal{FL})$.

2.2 The Language $\mathcal{L}(\mathcal{FL})$

$\mathcal{L}(\mathcal{FL})$ is a many-sorted first-order language. For the purposes of this paper, we assume two sorts: a sort $\mathcal{T}$ for time and a sort $\mathcal{F}$ for propositional fluents. In other work [6], an additional sort is used for actions. The language includes the predicate symbols *Holds* and *Occlude* , of type $\mathcal{T} \times \mathcal{F}$, and the predicate symbols $<$ and $\leq$ (interpreted as the usual "less than" and "less than or equal to" relations on natural numbers) of type $\mathcal{T} \times \mathcal{T}$, the equality predicate $=$, and the function symbols $+$, and $-$ (interpreted as the usual "plus" and "minus" functions on natural numbers) of type $\mathcal{T} \times \mathcal{T} \rightarrow \mathcal{T}$.

The numerals $0, 1, 2, \ldots$ and the symbols $t_0, t_1, \ldots$, will be used to denote constants of type $\mathcal{T}$ and the symbols $t_0, t_1, \ldots$ will be used to denote variables of type $\mathcal{T}$. We define the *set of temporal terms* to be the closure of the temporal variables and temporal constants of the language under the operators $+$ and $-$. A propositional fluent is a function of time with the boolean truth values as range. The symbols $f_1, f_2, \ldots$ will be used to denote variables of type $\mathcal{F}$. We assume an appropriate set of function symbols of proper arity for fluent names (e.g. *alive*, *at*). If F is a fluent name then a restricted fluent term is defined as a term with form $F(u_1, \ldots, u_n)$, where $u_1, \ldots, u_n$ are terms of sort *object*, where *object* is restricted to be either a variable or a constant term. Restricted fluent terms of arity 0 are called *fluent constants*. We define the *set of fluent terms* to be the union of fluent variables and restricted fluent terms.

An *atomic formula* is defined as any formula of the form $Holds(\mathrm{t}, \mathrm{f})$ or $Occlude(\mathrm{t}, \mathrm{f})$, where t is a temporal term and f is a fluent term. The set of formulas of $\mathcal{L}(FL)$ is defined as the closure of the atomic formulas under the boolean connectives (i.e. $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$) and the universal and existential quantifiers (i.e. $\forall$, $\exists$). In what follows, $\mathrm{t} < \mathrm{t}' < \mathrm{t}''$, $\mathrm{t} \leq \mathrm{t}' < \mathrm{t}''$, $\mathrm{t} < \mathrm{t}' \leq \mathrm{t}''$ and $\mathrm{t} \leq \mathrm{t}' \leq \mathrm{t}''$, stand for $\mathrm{t} < \mathrm{t}' \wedge \mathrm{t}' < \mathrm{t}''$, $\mathrm{t} \leq \mathrm{t}' \wedge \mathrm{t}' < \mathrm{t}''$, $\mathrm{t} < \mathrm{t}' \wedge \mathrm{t}' \leq \mathrm{t}''$ and $\mathrm{t} \leq \mathrm{t}' \wedge \mathrm{t}' \leq \mathrm{t}''$, respectively.

The intended interpretation for $\mathcal{T}$ is linear discrete time where $\mathcal{T}$ is considered isomorphic to the natural numbers. Since there is no axiomatization for time interpreted as the natural numbers, we either assume an interpreted language, settle for something less such as "integer-like, discrete flow of time with a first moment" which is axiomatizable [14], or assume a sound, but incomplete axiomatization of the flow of time. In practice, we will normally be using a specialized temporal constraint module for reasoning about time which is normally sound, but incomplete.

2.3 From $\mathcal{L}(SD)$ to $\mathcal{L}(FL)$

As stated in Section 2.1, the second step of the translation process uses macro-translation definitions to translate statements in Υ' into formulas in $\mathcal{L}(FL)$. We need a few preliminary definitions which will prove useful both here and in the definition of causal constraints in a later section. A *fluent formula* is any boolean combination of restricted fluent terms from $\mathcal{L}(FL)$. An *elementary scenario formula* is of the form $[t]\gamma$, where t is a temporal term in $\mathcal{L}(FL)$ and γ is a fluent formula. A *scenario formula* is any boolean combination of elementary scenario formulas. Let γ and δ denote fluent formulas and ϵ denote a restricted fluent term (possibly negated). Let C be any of the logical connectives $\wedge$, $\vee$, or $\rightarrow$. The following list of macro-translation definitions should suffice to provide the general idea:

$$[t](\delta \ C \ \gamma) \stackrel{def}{=} [t]\delta \ C \ [t]\gamma.$$

Any elementary scenario formula can be reduced to a boolean combination of elementary scenario formulas of the form $[t]\epsilon$.

$$
\begin{aligned}
[s,t]\delta &\stackrel{def}{=} \forall x.s \leq x \leq t \rightarrow [x]\delta \\
[s,t)\delta &\stackrel{def}{=} \forall x.s \leq x < t \rightarrow [x]\delta \\
[t]\epsilon &\stackrel{def}{=} \mathit{Holds}(t,\epsilon) \\
[s,t]\epsilon := T &\stackrel{def}{=} \mathit{Holds}(t,\epsilon) \\
&\quad \wedge \forall t_1(s < t_1 \leq t \rightarrow \mathit{Occlude}(t_1,\epsilon)) \\
[s,t]\epsilon := F &\stackrel{def}{=} \mathit{Holds}(t,\neg\epsilon) \\
&\quad \wedge \forall t_1(s < t_1 \leq t \rightarrow \mathit{Occlude}(t_1,\epsilon)) \\
\mathit{Holds}(t,\neg\epsilon) &\stackrel{def}{=} \neg \mathit{Holds}(t,\epsilon).
\end{aligned}
$$

The translation of Υ' in Example 2 into $\mathcal{L}(FL)$ using the translation rules is shown below:

Example 3

$$
\begin{aligned}
&\mathit{obs}0 \quad \mathtt{t}_0 = 0 \wedge \mathtt{t}_1 = 10 \\
&\mathit{obs}1 \quad \mathit{Holds}(\mathtt{t}_0, \mathit{alive}) \\
&\mathit{obs}2 \quad \neg\mathit{Holds}(\mathtt{t}_1, \mathit{alive}) \\
&\mathit{scd}1 \quad \mathit{Holds}(2, \mathit{loaded}) \rightarrow \\
&\qquad [\neg\mathit{Holds}(6, \mathit{alive}) \wedge \neg\mathit{Holds}(6, \mathit{loaded}) \wedge \\
&\qquad \forall t(2 < t \leq 6 \rightarrow \mathit{Occlude}(t, \mathit{alive})) \wedge \\
&\qquad \forall t(2 < t \leq 6 \rightarrow \mathit{Occlude}(t, \mathit{loaded}))]
\end{aligned}
$$

Note that although the labels are not part of the language of $\mathcal{L}(FL)$, they are retained. The labels are directly correlated with the partitioning of formulas used in the circumscription policy described in the next section. The following notation

$$\Gamma_C = \Gamma_{OBS} \cup \Gamma_{SCD} \cup \Gamma_{UNA} \cup \Gamma_T,$$

is used for a scenario description in $\mathcal{L}(FL)$, where Γ_{OBS} and Γ_{SCD} contain the observation and schedule statements in the scenario, Γ_T contains the axiomatization for the flow of time (when provided), and Γ_{UNA} contains the appropriate unique name axioms for the sorts $\mathcal{T}$ and $\mathcal{F}$. In the rest of the paper, we will use the convention of suppressing Γ_{UNA} and Γ_T, assuming they are provided with every theory. In addition, we will use the notation Γ_X, where X is an acronym such as OBS, for a finite set of formulas or their conjunction in contexts where this makes sense.

3 PMON Circumscription

In this section, we will describe the intuition behind the use of occlusion, introduce a Nochange Axiom, describe the filtered minimization technique, provide a circumscription policy which uses occlusion, the Nochange Axiom, and filtering, and show that any theory in the $\mathcal{K} - IA$ class of action scenarios is reducible to a first-order theory.

3.1 Occlusion

As we already mentioned in the introduction, the use of the occlusion concept and its representation in terms of the predicate *Occlude* has already proven to be quite versatile in providing solutions to a number of open problems associated with the representation of action and change. Although related to the use of an abnormality predicate together with an inertia assumption, there are some differences. The main difference is perspective. *Occlude* is used to provide a fine-grained means of excluding particular fluents at particular points in time from being subject to what are normally very strong inertia assumptions. In fact, in retrospect much of the progress made in solving a number of problems stemming from the original Yale Shooting Scenario has been the gradual relaxation of strict inertia in dealing with non-determinism, postdiction and in the current case, indirect effects and delayed effects of actions. It is the fine-grained use of *Occlude* together with the filtered minimization technique, where *Occlude* is minimized in only parts of theories, that contributes to the simplicity of the solutions. Filtering minimizes the need for complex minimization strategies. In fact, most of the time, the minimization policy involves little more than applying predicate completion to *Occlude* relative to part of a theory.

Recall the scenario description described in Section 2.3. Associated with each action type in a scenario is a subset of fluents that are potentially influenced by the action (those fluents in the right side of a rule). If the action has duration, then during its execution, it is not known in general what value the influenced fluents have. Since the action performance can potentially change the value of these fluents at any time, all that can generally be asserted is that at the end of the duration the fluent is assigned a specific value. To specify such behavior, the *Occlude* predicate is used in the definition of reassignment expressions which in turn are used as part of the definition of an action schema. *Occlude* serves the purpose of excluding certain features at certain time-points from the global inertia policy which we will soon introduce. In order to specify actions with durations and indeterminate effects of actions properly, it should be clear that fluents directly set by an action should be occluded during the execution of the action. In Lifschitz's recent terminol-

ogy, occluded fluents are simply frame-released fluents.

The predicate *Occlude* takes a time-point and a fluent as arguments. The definition for a reassignment expression $[t_1, t_2]\delta := T$ used in an action occurrence statement is[2]

$$Holds(t_2, \delta) \wedge \forall t(t_1 < t \leq t_2 \rightarrow Occlude(t, \delta)).$$

Referring to our previous example, it can be observed that the occlusion specification is automatically generated by the translation process from $\mathcal{L}(SD)$ to $\mathcal{L}(FL)$. Occlusion specifies what fluents may change at what points in time. The Nochange Axiom described next specifies when a fluent is *not* permitted to change value.

3.2 The PMON Circumscription Policy

Let Γ_{NCG} denote the following Nochange Axiom,

$$\forall f, t(Holds(t, f) \oplus Holds(t{+}1, f) \rightarrow Occlude(t{+}1, f)), \quad (1)$$

where the connective $\oplus$ is an abbreviation for the exclusive-or connective. The axiom states that if a fluent f is not occluded at $t+1$ then it can not change value from t to $t+1$. This axiom, together with the observation axioms will be used to filter potential histories of action scenarios.

Filtered preferential entailment is a technique originally introduced by Sandewall [16] for dealing with postdiction. The technique is based on distinguishing between different types of formulas in a scenario description and applying minimization to only part of the scenario, or different minimization policies to different parts of the scenario. In this particular case, we will distinguish between schedule statements Γ_{SCD} and the rest of the scenario. The idea is minimize the *Occlude* predicate relative to the schedule statements and then filter the result with the observation formulas, Γ_{OBS} and the nochange axiom Γ_{NCG}. The minimization policy generates potential histories where the potential for change is minimized. The potential histories are then filtered with the observations, which must hold in any valid history, and with the nochange axiom which filters out any spurious change not explicitly axiomatized by the actions. More formally, instead of using the policy,

$$\Gamma_{NCG} \wedge \Gamma_C \wedge \quad (2)$$
$$Circ_{SO}(\Gamma_{NCG} \wedge \Gamma_C(Occlude); Occlude),$$

where $Circ_{SO}$ denote standard second-order circumscription, PMON circumscription is defined using the policy,

$$\Gamma_{NCG} \wedge \Gamma_C \wedge Circ_{SO}(\Gamma_{SCD}(Occlude); Occlude). \quad (3)$$

Observe that the circumscription policy is surprisingly simple, yet at the same time assessed correct for the broad ontological class $\mathcal{K}$-*IA*. One simply minimizes the *Occlude* (frame-released) predicate while leaving *Holds* fixed in that part of the theory containing the action occurrences and then filters the result with the nochange (inertia) axiom and the observation axioms.

Although $Circ_{SO}(\Gamma_{SCD}(Occlude); Occlude)$ is a second-order formula, it can be shown that it is equivalent to a first-order formula using two results by Lifschitz [9], and the fact that *Occlude*-atoms only occur positively in Γ_{SCD}, or through the use of predicate completion. Details may be found in [2, 1]. In the following sections, we will show that this condition is satisfied even after PMON is extended for ramification.

4 Extending PMON for Ramification

Details regarding the work described so far can be found in previous publications by our group. We will now proceed to the main topic of this paper, that of extending PMON to PMON(RCs) in order to deal with indirect effects of actions. The ramification problem states that it is unreasonable to explicitly specify all the effects of an action in the action specification itself. One would rather prefer to state the direct effects of actions in the action specification and use the deductive machinery to derive the indirect effects of actions using the direct effects of actions together with general knowledge of dependencies among fluents, specified as domain constraints. The dependencies specified using domain constraints do not necessarily have to be based solely on notions of physical causality.

The idea is that there is a certain tiered precedence for change where change for fluents in a certain class are dependent on changes of fluents in another class but not vice-versa. Which fluents have precedence over others is of course a domain dependent call and that information must be provided in some manner, for example, either explicitly in terms of causal rules, or perhaps implicitly in terms of partitioning fluents in classes such as framed, frame-released and non-framed as is the case with [8], and using a particular minimization policy. A particular domain policy might be based on physical causality, where the precedence is for causes to have a stronger inertia quota than their dependencies. On the other hand, when explaining effects, one might reverse the precedence. A good example is the

[2] For $[t_1, t_2]\delta := F$, simply negate the *Holds* predicate.

domain constraint $light \equiv switch1 \vee switch2$ where causal flow is in the right-left direction, but one might equally well reverse the precedence for actions which turn a light on instead.

One of the difficulties in dealing with the ramification problem is the fact that a tension exists between solving the standard frame problem which requires minimizing change across the board, and attacking the ramification problem which requires relaxing minimization of change *just enough* to permit change for indirect effects, but only indirect effects that have a justification for changing. As mentioned above, it is still an open issue as to what policies such justifications should be based on. Physical causality is simply one of several reasons one might set up dependencies between fluents. Sandewall [19] and Thielscher [20] have recently begun analyzing different policies for justifying dependencies between fluents.

The basis of our solution is a straightforward encoding of relaxing minimization of change "just enough" for the indirect effects of actions. This will be done by introducing causal rules in $\mathcal{L}(SD)$ which provide a means of expressing the directionality of dependencies between fluents and defining their translations into $\mathcal{L}(FL)$ in terms of the *Occlude* predicate, which excludes the indirect effects from the nonchange constraint in the nochange axiom Γ_{NCG}. We will begin by distinguishing between two types of domain constraints, *causal constraints* and *acausal constraints*, and then specifying both their representations in $\mathcal{L}(SD)$ and translations into $\mathcal{L}(FL)$, in terms of the *Occlude* predicate.

4.1 Causal and Acausal Constraints

In order to express causal constraints, we begin by defining causal relation expressions in $\mathcal{L}(SD)$.

Definition 1 A *causal relation* is an expression in $\mathcal{L}(SD)$ with the following form,

$$[t]\delta \gg [s]\gamma,$$

where both t and s are temporal terms in $\mathcal{L}(FL)$, $t \leq s$, and δ and γ are fluent formulas. □

The intended meaning of a causal relation is "if the fluent formula δ is true at time-point t, then the fluent formula γ must be true at time-point s, and a change in γ due to this rule is legal w.r.t the nochange premise".

Definition 2 A *causal constraint* is an expression in $\mathcal{L}(SD)$ with the following form,

$$Q(\alpha \rightarrow [t]\delta \gg [s]\gamma) \text{ or } Q([t]\delta \gg [s]\gamma),$$

where α is a scenario formula in which for any temporal term t' in α,$t' \leq t$, and Q is a sequence of quantifiers binding free variables of sorts $\mathcal{F}, \mathcal{T}$. □

We call α the *precondition* for the causal constraint. The use of preconditions permits the representation of context dependent dependencies among fluents. Note also, that because the formalism uses explicit time and both t and s may refer to different time-points, it is straightforward to represent delayed effects of actions using causal constraints. We will demonstrate this in the examples which follow in the next section. Causal constraints will be labeled with the prefix *csc* in scenario descriptions in $\mathcal{L}(SD)$. The following expressions provide examples of conditionalized and non-conditionalized causal constraints:

$$\text{csc} \quad \forall t([t]underwater \rightarrow ([t]breathing \gg [t + \mathtt{t}_0]\neg alive))$$
$$\text{csc} \quad \forall t([t]\neg alive \gg [t]\neg walking),$$

Acausal constraints describe relations between fluents without encoding any preference for dependency ordering. In a sense, a special class of acausal constraints is not really necessary. Technically, they could be defined as observations in a scenario which are observed at all time-points, but conceptually there is a difference.

Definition 3 An *acausal constraint* is an expression in $\mathcal{L}(SD)$ which is an arbitrary quantified scenario formula. □

Acausal constraints will be labeled with the prefix "acc" in $\mathcal{L}(SD)$. We call an acausal constraint where all fluents have the same temporal term, a *static domain constraint*, while those with several terms will be called *transition constraints*. The following expressions provide examples of static and transition acausal constraints, respectively:

$$\text{acc} \quad \forall t([t]\neg black \vee [t]\neg white)$$
$$\text{acc} \quad \forall t([t]\neg alive \rightarrow [t+1]\neg alive).$$

4.2 Translation into $\mathcal{L}(FL)$

In the previous section, we defined a number of different domain constraint types in the surface language . We will now provide a translation into $\mathcal{L}(FL)$ and show that our informal intuitions regarding dependency preferences among constraints are formally encoded into $\mathcal{L}(FL)$. Note that the only new symbol introduced when defining our domain expressions is the symbol $\gg$. The following macro-translation definition provides the proper translation for this relation.

Definition 4

$$[t]\delta \gg [s]\gamma \stackrel{def}{=} [([t]\delta \rightarrow [s]\gamma) \wedge \quad (4)$$
$$(([t-1]\neg\delta \wedge [t]\delta) \rightarrow [s]X(\gamma))], \quad (5)$$

where $t \leq s$ and $X(\gamma)$ denotes the occlusion of all restricted fluent terms in γ at s.[3] □

This intermediate formula is then translated into $\mathcal{L}(FL)$ in a manner similar to that described in the example in Section 2.3.

Example 4 The following causal constraint expression specified in $\mathcal{L}(SD)$,

$$\text{csc} \qquad \forall t[t](\neg alive \gg \neg walking)^4,$$

is translated into the following $\mathcal{L}(FL)$ formula,

$$\begin{aligned}\text{csc} \quad & \forall t\{ \\ & (\neg Holds(t, alive) \rightarrow \neg Holds(t, walking)) \wedge \\ & (Holds(t-1, alive) \wedge \neg Holds(t, alive) \rightarrow \\ & \qquad Occlude(t, walking)).\}\end{aligned}$$

□

The intuition behind the translation in Definition 4 is as follows: the first part of Definition 4, (4), represents the actual causal dependency. $[t]\delta \rightarrow [s]\gamma$ forces γ to be true if δ is true. The second part (5) simply justifies the changes caused by the first part, but only in the appropriate causal direction. $([t-1]\neg\delta \wedge [t]\delta) \rightarrow [s]Occlude(\gamma)$, states that a change in γ caused by the rule is legal w.r.t the nochange axiom.

4.3 PMON(RCs) Circumscription

The language of scenario descriptions $\mathcal{L}(SD)$ has been extended for causal and acausal constraints and additional macro-translation definitions have been introduced which permit the translation of scenario descriptions with domain constraints into $\mathcal{L}(FL)$. The final step will be to extend the circumscription policy used in PMON to accommodate these new changes.

Let Γ_{ACC} and Γ_{CSC} denote the translations of the acausal and causal constraints into $\mathcal{L}(FL)$, respectively. An action scenario Γ_C is now defined as $\Gamma_C = \Gamma_{FIL} \cup \Gamma_{CHG}$ where,

$$\Gamma_{FIL} = \Gamma_{NCG} \cup \Gamma_{OBS} \cup \Gamma_{ACC} \cup \Gamma_{UNA} \cup \Gamma_T$$

$$\Gamma_{CHG} = \Gamma_{SCD} \cup \Gamma_{CSC}.$$

The new circumscription policy is defined as

$$\Gamma_{FIL} \wedge Circ_{SO}(\Gamma_{CHG}(Occlude); Occlude), \quad (6)$$

Note that since Γ_{CHG} contains no negative occurrences of *Occlude*, and all other predicates are fixed, any action scenario in PMON(RCs) is provably reducible to a logically equivalent first-order theory. The first-order reduction applies to the extended scenarios without change.

5 Examples

In this section, we will consider two examples from the literature, the latter slightly modified from the original. These examples should help in acquiring both a conceptual and technical understanding of PMON(RCs).

Example 5 The walking turkey problem is a well-known ramification scenario and relatively straightforward to encode and solve using causal constraints. We will require one causal constraint stating that dead turkeys do not walk. One ramification of shooting a turkey is that it no longer walks and our theory should entail this indirect effect. The following action scenario description in $\mathcal{L}(SD)$ describes the walking turkey problem:

$$\begin{aligned} obs1 \quad & [0]walking \\ ac1 \quad & [2,4]Shoot \\ acs1 \quad & [s,t]Shoot \rightsquigarrow [s,t]alive := F \\ csc1 \quad & \forall t[t]\neg alive \gg \neg walking \end{aligned}$$

The corresponding translation into $\mathcal{L}(FL)$ is,

$$\begin{aligned} obs1 \quad & Holds(0, walking) \\ scd1 \quad & \neg Holds(4, alive) \wedge \forall t(2 < t \leq 4 \rightarrow \\ & \quad Occlude(t, alive)) \\ csc1 \quad & \forall t[(\neg Holds(t, alive) \rightarrow \neg Holds(t, walking)) \wedge \\ & \quad (Holds(t-1, alive) \wedge \neg Holds(t, alive) \rightarrow \\ & \quad Occlude(t, walking))] \end{aligned}$$

The scenario is partitioned as:

$\Gamma_{FIL} = \Gamma_{NCG} \cup \Gamma_{OBS} \cup \Gamma_{ACC} \cup \Gamma_{UNA} \cup \Gamma_T$, where

$\Gamma_{OBS} = \{obs1\}, \Gamma_{UNA} = \{alive \neq walking\}, \Gamma_{ACC} = \{\}.$

$\Gamma_{CHG} = \Gamma_{SCD} \wedge \Gamma_{CSC}$, where

$$\Gamma_{SCD} = \{scd1\}, \Gamma_{CSC} = \{csc1\}.$$

[3] For example, if γ is $\epsilon_1 C \epsilon_2$, C is a logical connective, then $[s]X(\gamma)$ is $Occlude(s, \epsilon_1) \wedge Occlude(s, \epsilon_2)$.

[4] In the rest of the paper $[t](\delta \gg \gamma)$ will often be used instead of $[t]\delta \gg [t]\gamma$.

Circumscribing *Occlude* in Γ_{CHG} results in the following definition which can be generated using either our algorithm [5], or standard predicate completion:

$$\forall t, f(Occlude(t,f) \leftrightarrow \quad ((f = alive \wedge 2 < t \leq 4) \vee (f = walking \wedge Holds(t-1, alive) \wedge \neg Holds(t, alive))))$$

For readability, we list the complete translation of the scenario in $\mathcal{L}(FL)$, with the definition of *Occlude* derived via circumscription:

$$\begin{aligned}
&\forall f, t(Holds(t,f) \oplus Holds(t+1,f) \rightarrow \\
&\quad Occlude(t+1,f)) \wedge \\
&Holds(0, walking) \wedge \\
&alive \neq walking \wedge \\
&\neg Holds(4, alive) \wedge \\
&\forall t[(\neg Holds(t, alive) \rightarrow \neg Holds(t, walking)) \wedge \\
&\forall t, f(Occlude(t,f) \leftrightarrow ((f = alive \wedge 2 < t \leq 4) \vee \\
&\qquad (f = walking \wedge Holds(t-1, alive) \wedge \\
&\qquad \neg Holds(t, alive))))
\end{aligned}$$

There are two classes of preferred models for this scenario due to the fact that the shoot action has duration and its effects may occur at time point 3 or 4:

$$[0,2]alive \wedge walking$$
$$[3,\infty]\neg alive \wedge \neg walking$$

and

$$[0,3]alive \wedge walking$$
$$[4,\infty]\neg alive \wedge \neg walking$$

Example 6 The extended stuffy room problem is based on the original problem due to Winslett [21]. We extend it by introducing a box which requires the use of chained causal rules, and by encoding a delayed effect of an action by using one of the causal rules.

In this scenario, there are two causal constraints. The first asserts that *duct*2 is blocked if something is put on top of it (*loc*2). The second asserts that the room gets stuffy two time-points after both ducts are blocked. Between time-points 2 and 5 a box is moved to location *loc*2. The action scenario is represented in $\mathcal{L}(SD)$ as:

$$\begin{aligned}
&obs && [0](blocked(duct1) \wedge \neg blocked(duct2) \wedge \\
&&& at(box, loc1) \wedge \neg stuffy) \\
&ac && [2,5]move(box, loc1, loc2) \\
&acs && \forall x, l1, l2([s,t]move(x,l1,l2) \rightsquigarrow \\
&&& [s,t]at(x,l1) := F \wedge [s,t]at(x,l2) := T) \\
&csc1 && \forall t, x[t](at(x, loc2) \gg blocked(duct2)) \\
&csc2 && \forall t([t](blocked(duct1) \wedge blocked(duct2)) \gg \\
&&& [t+2]stuffy)
\end{aligned}$$

Translating into $\mathcal{L}(FL)$ and circumscribing, results in the following theory:

$$\begin{aligned}
&Holds(0, blocked(duct1)) \wedge \\
&\neg Holds(0, blocked(duct2)) \wedge \\
&Holds(0, at(box, loc1)) \wedge \\
&\neg Holds(0, stuffy) \wedge \\
&\neg Holds(5, at(box, loc1)) \wedge \\
&Holds(5, at(box, loc2)) \wedge \\
&\forall t, x(Holds(t, at(x, loc2)) \rightarrow \\
&\quad Holds(t, blocked(duct2))) \wedge \\
&\forall t(Holds(t, blocked(duct1)) \wedge \\
&\quad Holds(t, blocked(duct2)) \rightarrow \\
&\quad Holds(t+2, stuffy)) \wedge \\
&\forall f, t(Holds(t,f) \oplus Holds(t+1,f) \rightarrow \\
&\quad Occlude(t+1,f)) \wedge \\
&\forall t, f(\\
&\quad Occlude(t,f) \leftrightarrow \\
&\quad (2 < t \leq 5 \wedge f = at(box, loc1)) \vee \\
&\quad (2 < t \leq 5 \wedge f = at(box, loc2)) \vee \\
&\quad \forall x(\neg Holds(t-1, at(x, loc2)) \wedge \\
&\quad\quad Holds(t, at(x, loc2)) \wedge \\
&\quad\quad f = blocked(duct2)) \vee \\
&\quad (\neg[Holds(t-3, blocked(duct1)) \wedge \\
&\quad\quad Holds(t-3, blocked(duct2))] \wedge \\
&\quad Holds(t-2, blocked(duct1)) \wedge \\
&\quad Holds(t-2, blocked(duct2)) \wedge \\
&\quad f = stuffy) \\
&)
\end{aligned}$$

In addition, the unique name axioms are included, but not listed here. The theory correctly entails that if something is placed on top of *duct*2 while *duct*1 is blocked, then the room will become stuffy two time-points later. In the current axiomatization, the room would remain stuffy forever even after one of the ducts became free. This is because the nochange axiom prevents the room from becoming unstuffy without reason. In the current theory, there is no axiom which states otherwise. A rule stating that the room is stuffy only when both ducts are blocked can be added with-

out difficulty:

$$csc3 \quad \forall t([t]\neg(blocked(duct1) \wedge blocked(duct2)) \gg [t+2]\neg stuffy).$$

This example demonstrates both the use of casual chaining and how causal constraints can be used to specify delayed effects of actions. □

6 Causal constraints and Stratification

Causal cycles without delay cause spontaneous or non-grounded change, as the example below illustrates.

Example 7

$$\begin{array}{ll} obs1 & [0]\neg\alpha \\ csc1 & \forall t[t](\alpha \gg \alpha) \end{array} \qquad (7)$$

Unfortunately, this allows a model where α holds at time-point 1, without any outside cause. Where intuitively we would assume that $\neg\alpha$ persists, because there is no reason for it not to, the causal rule introduces unsupported occlusion, blocking this preferred entailment.□

Lin solves this problem by demanding that the causal constraints are stratified, the following definition is from Lin [11], it has been slightly changed for our terminology.

Definition 5 The csc's are stratified if there are no fluents $F_0, F_1, ..., F_n$ such that $F_0 \rightarrow F_1 \rightarrow ... \rightarrow F_n \rightarrow F_0$, where for any fluents F and F', $F' \rightarrow F$ if there is a causal relation such that F' appears in the left hand side of the $\gg$ sign, and F appears in the right hand side of the causal relation. □

Lin requires this definition, not only to rule out non-grounded change, but to avoid cycles or recursion in his causal rules, which prevents the use of Clark's completion when reducing 2nd-order theories. Note that we are never in this position because the translation of causal constraints in our formalism only generates positive occurrences of *Occlude* and the use of restricted fluent terms prevents any recursion in *Occlude* even if negative occurrences were generated.

One way to view stratification is to think in terms of positive and negative recursion. *Positive recursion* occurs when F_0 has the same sign at the beginning and end of a chain like that in Definition 5. A typical positive recursion is shown in Example 7. On the other hand, *Negative recursion* occurs when F_0 has different signs at the beginning and end of a chain. The rules $\alpha \gg \beta, \beta \gg \neg\alpha$ together provide an example of negative recursion. The current stratification definition can be viewed as a means of ruling out both types of recursion.

Although one might have a difficult time finding an example where negative recursion makes sense with causal rules, allowing it in our formalism does not cause any technical problems. For example, the first part of the translation of $\forall t[t](\alpha \gg \neg\alpha)$ is $\forall t([t]\alpha \rightarrow [t]\neg\alpha$ which is equivalent to $\forall t[t]\neg\alpha$ and the lhs of the second part of the translation is always false, so no additional occlusion assertions are generated.

As shown in the example, positive recursion when allowed, may result in unintuitive and unintended models, where a causal constraint in some sense triggers itself. On the other hand, if we restrict ourselves to causal rules which may generate positive recursive cycles, but where the cause occurs strictly before the effect, then the presence of positive cycles is technically unproblematic.

Consequently, PMON(RCs), does not require a stratification definition as restrictive as the one above, but the set of non-delayed causal constraints (constraints in which $t = s$ in Definition 2) must not contain positive recursion.

The requirement that we exclude all types of non-delayed positive recursive causal constraints when using our approach is unfortunate because this would rule out using a number of useful domain constraints in both directions such as $alive \leftrightarrow \neg dead$. If we encode this formula using causal constraints in an attempt to permit precedence in both directions the result would be positive non-delayed cycles, which we have already stated generate unintuitive models. Let's take another example. Suppose we would like to encode the constraint $lamp \leftrightarrow switch_1 \vee switch_2$, so it may be used in the causal direction (right-to-left) and for explanation in the other direction (left-to-right) using causal rules. The following rules would be needed:

$$\begin{array}{ll} csc1 & \forall t[t](la \gg (sw_1 \vee sw_2)) \\ csc2 & \forall t[t](\neg la \gg \neg(sw_1 \vee sw_2)) \\ csc3 & \forall t[t]((sw_1 \vee sw_2) \gg la) \\ csc4 & \forall t[t](\neg(sw_1 \vee sw_2) \gg \neg la), \end{array}$$

Unfortunately, these are positive recursive and would not do the job. In other words, it is very difficult to use the causal rule approach in general when we are dealing with constraints without causal interpretations. Sandewall [19] discusses this in detail. This is obviously a limiting feature of the current approach and approaches like ours.

7 Related work

7.1 Lin

Previously, we mentioned that if one disregards failed actions and qualification when comparing Lin's recent proposals for dealing with indirect effects of actions and indeterminate actions([11]), then there are striking similarities between these proposals and the original and extended versions of PMON. In fact, the *Cause* predicate and minimization policy used by Lin are virtually analogous with the use of the *Occlude* predicate and minimization policy used in both the original PMON and its extension PMON(RCs). Assuming familiarity with [11], let's briefly compare the two frameworks. We'll begin with syntax and then discuss minimization policy.

Compare Lin's frame axiom:

$$Poss(a,s) \rightarrow \{ \quad \neg(\exists v)Caused(p,v,do(a,s)) \rightarrow [Holds(p,do(a,s)) \leftrightarrow Holds(p,s)]\},$$

which is equivalent to

$$Poss(a,s) \rightarrow \{ \quad [Holds(p,do(a,s)) \oplus Holds(p,s)] \rightarrow (\exists v)Caused(p,v,do(a,s))\}$$

with the nochange axiom in PMON:

$$\forall f,t(Holds(t,f) \oplus Holds(t{+}1,f) \rightarrow Occlude(t{+}1,f)).$$

$Poss(a,s)$ has to do with failed actions so this can be disregarded.

In Lin's example, a typical causal domain constraint is represented as,

$$up(L1,s) \wedge up(L2,s) \rightarrow Caused(open,true,s)$$

An additional rule is used to transfer change in the *Caused* context back to the *Holds* context:

$$Caused(p,true,s) \rightarrow Holds(p,s),$$

which can be used to rewrite the previous causal constraint to the equivalent,

$$\forall s \quad ([up(L1,s) \wedge up(L2,s) \rightarrow Holds(open,s)] \wedge [up(L1,s) \wedge up(L2,s) \rightarrow Caused(open,true,s)]),$$

which would be quite similar to our representation of a causal constraint as,

$$\forall t \quad ([[t]up(L1) \wedge [t]up(L2) \rightarrow [t]open] \wedge [[t-1]\neg(up(L1) \wedge up(L2)) \wedge [t](up(L1) \wedge up(L2)) \rightarrow [t]Occlude(open)].$$

Lin describes his minimization method as follows:

1. Start with a theory T that includes all the effect axioms and state constraints.
2. Minimize *Caused* in T. Let T' be the resulting theory.
3. Add to T' the frame axiom(10). Let T'' be the resulting theory.
4. Maximize *Poss* in T'' to obtain the final action theory.

Again, we can disregard step 4 in the comparison, which leaves us with steps 1-3. Disregarding what we place in parentheses for the moment, PMON's original minimization policy is,

1. Start with a theory T that includes all the schedule axioms Γ_{SCD}, (and state constraints Γ_{csc}).
2. Minimize *Occlude* in T. Let T' be the resulting theory.
3. Add to T' the nochange axiom Γ_{NCG} and observations Γ_{OBS}. Let T'' be the resulting theory.

To acquire the extended minimization policy for PMON(RCs), simply remove the parentheses in step 1.

One difference between Lin's approach and ours (and we are sure there may be more) is that Lin keeps fluents *Caused* as long as the constraint is active, while we only *Occlude* when the actual change of value takes place. The nochange axiom takes care of the continued behavior.

7.2 Thielscher

Another interesting approach to ramification is suggested by Thielscher [20]. He regards the direct effects obtained after firing an action merely as a preliminary approximation to the resulting situation which is then computed by performing additional post-processing steps that generate indirect effects relative to domain constraints specified as causal rules. For each action, there is a state representing the direct effects of the action followed by a sequence of states representing the progression towards a state that is legal w.r.t the domain constraints[5]. In order to "fire" causal rules in the right order and direction, additional knowledge about the directionality of causation (*influence relationships*) has to be specified. These influence relationships together with the domain constraints are compiled into

[5] In the rest of this section we will call sequences like these ramification sequences

causal relations, which are then used to progress between states in the ramification sequence.

Thielscher does not require stratified theories, this is possible since each application of a causal relation leads to a new state, so non-grounded change is ruled out by definition.

Thielscher's approach, although very promising, has a number of limitations as regards expressiveness. For example, all actions must be deterministic and non-deterministic domain constraints sometimes lead to unintuitive results. For example the causal constraint $light \gg sw_1 \vee sw_2$ if represented in Thielscher's approach using the causal rules, $\neg s_1 : l \rightsquigarrow s_2$ and $\neg s_2 : l \rightsquigarrow s_1$, and these are applied sequentially, then it appears that turning on the *light* would not result in any states where both switches would be on. It also appears that causal rules leading to negative recursion could lead to a situation where no stable state is generated, although this problem may be dealt with using the logic programming technique which implements his formal specification.

7.3 Sandewall

In [19], Sandewall presents the *transition cascade semantics* which is an extension of previous work with the Features and Fluents framework. His approach to providing an underlying semantics for dealing with various types of ramification has some similarities with Thielscher's approach. Like Thielscher he considers the direct effects of an action as an approximation to the actual resulting situation. Actions are specified using an *action invocation relation* $D(E, r, r_1)$, where E is an action, r is the state where the action E is invoked, and r_1 is the new state where the instrumental part of the action has been executed. After the action has been fired, a binary, non-reflexive *causal transition relation* C is used to construct a ramification sequence $r_1, r_2, ..., r_k$ where $C(r_{i,i+1})$ for every i between 1 and $k-1$, and r_k is a stable state. r_k is considered as the result state of the action E in situation r. The set of result states is denoted $N(E, r)$. The idea is that result states contain both direct and indirect effects of an action occurrence.

The concept of *respectful* action systems is also introduced where any fluent in a ramification sequence may only change value once. This constraint, in effect, rules out negative recursion, but only with respect to each action. No action may lead to a ramification sequence where a fluent changes value more than once. Domain constraints may still contain negative recursion, but these cycles cannot be reached.

Sandewall's work is relatively new, but we believe the extended underlying semantics he proposes will be a useful tool for assessing the correctness of PMON(RCs) and determining what classes of action scenarios the logic can be shown correct. His current analysis of existing proposals should also provide a direct means of comparing our work with these other approaches.

8 Conclusion

We feel that PMON and it's extension PMON(RCs) have a number of advantages over other formalisms for specifying action scenarios. They use explicit time in terms of a linear metric time structure which allows one to specify actions with duration, delayed effects of actions, and the incomplete specification of timing and order of actions in a straightforward manner. We've argued that there is a simple and intuitive surface language for describing scenarios and that for this particular class of scenario descriptions, we can reduce the circumscribed scenarios to the first-order case algorithmically.

On the other hand, we have not claimed that this logic is suitable for all classes of problems, nor that it has solved the frame and ramification problems in total, whatever they may be defined as being for the moment. For future work, we would like to assess just what class of scenarios this particular logic functions properly for, using the Features and Fluents framework. Finally, we hope that the comparisons we have made with other formalisms contribute towards progressing in a forward direction by building not only on our own work, but also on the work of others.

References

[1] P. Doherty. Notes on PMON Circumscription. Technical report, Department of Computer and Information Science, Linköping University, Sweden, 1994.

[2] P. Doherty. Reasoning about Action and Change using Occlusion. In *Proceedings of the 11th European Conference on Artificial Intelligence, Aug. 8-12, Amsterdam*, pages 401–405, 1994.

[3] P. Doherty and W. Łukaszewicz. Circumscribing Features and Fluents. In D. Gabbay and H.J. Ohlbach, editors, *Proceedings of the 1st International Conference on Temporal Logic*, volume 827 of *Lecture Notes in Artificial Intelligence*. Springer, 1994.

[4] P. Doherty and P. Peppas. A comparison between two approaches to ramification: PMON(R) and $\mathcal{AR}_0$. In *Proceedings of the 8th Australian Joint Conference on Artificial Intelligence*, 1995.

[5] P. Doherty, W. Łukaszewicz, and A. Szałas. Computing Circumscription Revisited: Preliminary Report. In *Proceedings of the 14th Int'l Joint Conference on Artificial Intelligence*, volume 2, pages 1502–1508, 1995. Extended version to appear in Journal of Automated Reasoning.

[6] P. Doherty and W. Łukaszewicz and A. Szalas. Explaining Explanation Closure. In *Proc. ISMIS-96* , 1996.

[7] J. Finger. *Exploiting Constraints in Design Synthesis.* PhD thesis, Department of Computer Science, Stanford University, Stanford, CA, 1986.

[8] G.N. Kartha and V. Lifschitz. Actions with Indirect Effects (Preliminary Report). In *International Conference on Principles of Knowledge Representation and Reasoning*, 1994.

[9] V. Lifschitz. Pointwise Circumscription. In M. Ginsberg, editor, *Readings in Non-monotonic Reasoning.* Morgan Kaufmann, 1988.

[10] V. Lifschitz. Frames in the Space of Situations. *Artificial Intelligence*, 46:365–376, 1990.

[11] F. Lin. Embracing Causality in Specifying the Indirect Effects of Actions. *In Proceedings of 14th International Joint Conference on Artificial Intelligence*, 1995.

[12] F. Lin. Embracing Causality in Specifying the Indeterminate Effects of Actions. *Proc. AAAI'96*, 1996.

[13] F. Lin and R. Reiter. State Constraints Revisited. *Journal of Logic and Computation, Special Issue on Actions and Processes*, 1994.

[14] D. Gabbay and I. Hodkinson and M. Reynolds. Temporal Logic, Mathematical Foundations and Computational Aspects, Vol. 1. Oxford University Press, 1994.

[15] N. McCain and H. Turner. A Causal Theory of Ramifications and Qualifications. *In Proceedings of 14th International Joint Conference on Artificial Intelligence*, 1995.

[16] E. Sandewall. Filter Preferential Entailment for the Logic of Action in almost Continuous Worlds. *Proc. of IJCAI'89*, Morgan Kaufmann, 1989.

[17] E. Sandewall. *Features and Fluents. A Systematic Approach to the Representation of Knowledge about Dynamical Systems. Volume I.* Oxford University Press, 1994.

[18] E. Sandewall. Systematic Comparison of Approaches to Ramification using Restricted Minimization of Change. Technical report, Department of Computer and Information Science, Linköping University, Sweden, 1995.

[19] E. Sandewall. Assessments of ramification methods that use static domain constraints In *International Conference on Principles of Knowledge Representation and Reasoning* Morgan Kaufmann, 1996.

[20] M. Thielscher. Computing Ramifications by Postprocessing. In C. S. Mellish, editor, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1994–2000, Montreal, Canada, August 1995. Morgan Kaufmann.

[21] M. Winslett. Reasoning about actions using a possible models approach. In *National Conference on Artificial Intelligence*, 1988.

Assessments of ramification methods that use static domain constraints

Erik Sandewall
Department of Computer and Information Science
Linköping University
S-58183 Linköping, Sweden
erisa@ida.liu.se, http://www.ida.liu.se/~erisa

Abstract

The paper analyzes and compares methods for ramification, or for update, which are based on minimization of change while respecting static domain constraints. The main results are:

- An underlying semantics for propagation oriented ramification, called *causal propagation semantics*
- *Comparative assessments* of a number of previously proposed entailment methods for ramification, in particular, methods based on minimization of change after partitioning of the state variables (fluents). This includes the methods previously proposed by del Val and Shoham, by Kartha and Lifschitz, and by our group.
- Assessment of two *ranges of sound applicability* for one of those entailment methods, MSCC, relative to causal propagation semantics.
- Identification of an essential feature for minimization-based approaches to ramification, namely *changeset-partitioning.*

1 TOPIC AND APPROACH

1.1 THE RAMIFICATION PROBLEM

The *ramification problem* for reasoning about actions and change has been described as follows by Ginsberg and Smith, [6]:

> The difficulty is that it is unreasonable to explicitly record all of the consequences of an action, even the immediate ones. For example [...] For any given action there are essentially an infinite number of possible consequences that depend upon the details of the situation in which the action occurs.

More concretely, the difference between ramification and the simpler case of strict inertia is that strict inertia assumes action laws to specify *all* the effects of the action explicitly, whereas in ramification the action law only needs to specify *some of* the effects; the others are to be deduced. On the other hand, the logic cannot do magic; those other effects must also be declared somehow. One can think of two specific reasons why ramification is useful:

1. In order to avoid duplication of the same information on the consequent side of several action laws. Suppose, for a trivial example, that both *alive*(x) and *dead*(x) occur as state variables. Under strict inertia, every action that causes a change in *alive*(x) must also specify the corresponding change in *dead*(x).
2. In order to avoid configuration-dependent action laws. Suppose the scenario world consists of a number of interconnected objects, where a change in one of them may cause changes in neighboring objects, and different scenarios in the same world contain different configurations. If the same action law is going to be used in all scenarios, it must apply regardless of the choice of configuration, and then it is not possible to represent the secondary changes explicitly in the action law.

We shall refer to these as the *redundancy* reason and the *propagation* reason for ramification, respectively.

1.2 CURRENT APPROACHES

Most solutions which have been proposed for the ramification problem require action laws to specify the most significant effects of the action, and to rely on *domain constraints* for specifying additional changes that are due to the action. However, the imposition of domain constraints alone is not sufficient. Two courses have been taken. *Minimization based approaches* assume that the total set of changes in state variables that result from the action, consists of those changes that are explicitly stated in the action law, and a min-

imal set of other changes while satisfying the domain constraints. *Causation oriented approaches* require instead that some information is available about how changes in one state variable may "cause" changes in another state variable, and accept those models which satisfy action laws and domain constraints, where only correctly "caused" changes are present. Domain-constraint and influence information are sometimes integrated, sometimes separated.

The *update problem* is in principle the same problem: given a world state before an action, some information about the world state after the action, and certain general restrictions on permissible world states, which are the possible or preferred world states resulting from the action? The main difference is one of presentation: the ramification problem is presented in the context of reasoning about actions and change, where the actions themselves as well as their timing and sequencing is explicit in the logical language being used. For the update problem, by contrast, one only considers one update, and leaves aside the question of timing and sequencing of the multiple actions.

Within the general frameworks of minimization based and causation oriented approaches, there are a number of different specific proposals which have been advanced by different authors, and which differ in their details. So far, these methods have only been motivated on the basis of (1) intuitive and informal reasons why a particular method is the "natural" one, and (2) selected test examples where competing methods do not give intended results.

In earlier articles [17, 18] we have argued that this kind of evidence is not sufficient, and that proposed methods for nonmonotonic common-sense reasoning should be *assessed* with respect to an *underlying semantics* which formally defines the intended conclusions for a class of scenario descriptions. We have also performed such assessment of the range of applicability for about one dozen entailment methods dealing with strict inertia, that is, persistence without ramification. Yi, in [25], has generalized the underlying semantics to allowing concurrent actions, and has extended the assessment of range of applicability to one method for this case.

1.3 VARIETIES OF ASSESSMENT IN THE CASE OF RAMIFICATION

The present paper describes the first assessment results for current approaches to ramification. It turns out that the assessment methodology has to be revised in this case, for the following reasons. In the case of strict inertia, it was possible to obtain the range for completeness, that is, the range where the set of selected models *equals* the set of intended models. Ramification appears to be more difficult, and with the underlying semantics proposed below we have only been able to obtain certain results on the range of soundness, that is, the range where the set of selected models *contains* the set of intended models.

For many practical purposes, soundness is good enough. Methods which are sound but not complete involve of course a certain loss of information, but information is lost anyway in several other ways in any practical system. However, if we merely have soundness, then it is also of interest to know something about how well or how badly the set of selected models approximates the set of intended models.

For this reason, we propose to complement the *soundness assessment* with a *comparative assessment* where entailment methods are compared pairwise with respect to the extent of the model sets that they select. The present paper begins with an overview of minimization based entailment methods for ramification. Then follow the results on comparative assessment, then the proposed underlying semantics which is called *causal propagation semantics*, and finally some results on assessment of range of sound applicability wrt causal propagation semantics. This ordering is natural since the comparative assessment is technically easier to follow, and since it has been performed for a larger number of entailment methods.

The comparative assessment results are based on a more extensive departmental technical report [20] containing the full formal machinery and the full proofs. The formal aspects have previously been summarized in [19], although the main topic of that article was different from the present one.

2 PROS AND CONS OF CURRENT APPROACHES

2.1 PLAIN MINIMIZATION OF CHANGE

The original method, plain minimization of change, is originally due to Winslett [24], at least in the context of the A.I. literature. The idea is simple: identify a set of *static domain constraints*, that is, restrictions on the possible states of the world. For a given action, the effects specified by the action law are obligatory; a minimal set of additional changes is selected so that the domain constraints are satisfied.

It was observed already by Winslett that plain minimization of change sometimes obtains unintended results. She suggested to solve this by imposing a priority ordering on the state variables, but this line of research has apparently not been pursued since. The following approaches have been developed instead.

2.2 CATEGORIZATION OF STATE VARIABLES

Categorization of state variables is a fairly simple modification of plain minimization of change: instead of establishing certain changes in the world, and then minimizing changes across all (other) state variables, one divides the state variables into two or more classes which are treated differently with respect to minimization.

One instance of this concept was the introduction of *occlusion* by Sandewall [15, 17], also represented by the causal explanation predicate of Persson and Staflin [13], the *persists* predicate in [14], the *release* operation of Kartha and Lifschitz [8] and the *persists* predicate of del Val and Shoham in [4]. In the simplest case, one separates the state variables into two classes A and B, where the action laws specifying the effects of an action only specify changes in class A, whereas domain constraints are allowed to refer to state variables in both classes. Then, change is minimized for state variables in class B, whereas those in class A are allowed to vary freely to the extent permitted by the action laws.

Another categorization method is to single out some state variables as *dependent* on others, and to stipulate that minimization of change does not apply to dependent state variables. The definitions of the dependencies are part of the domain constraints. This approach was first proposed by Lifschitz as a distinction between frame and no-frame fluents [10].

2.3 CAUSATION ORIENTED APPROACHES

Plain minimization of change is symmetric: a constraint of the form $a \rightarrow b$, if imposed on a state where a is true and b is false, can be satisfied either by making a false, or by making b true. This does not always correspond to the intentions in the application at hand, since sometimes the influence applies only one way.

The distinction between dependent and independent state variables is often sufficient for dealing with this problem. However, Thielscher [23] has shown an example, involving relays, where the distinction between dependent and independent variables is not sufficient. In his paper, and in concurrent papers by other authors [12, 11], a "causal" approach is proposed. The basic idea is to express *causal dependencies*, which say that change in one state variable necessitates or enables a change in another state variable. Similar methods have been proposed earlier by Cordier and Siegel [2, 3] and by Brewka and Hertzberg [1]. Recently, Gustafsson and Doherty [7] have showed how a causation oriented approach can be reduced to a categorization approach without change minimization.

The fundamental limitation of current causal approaches is that they assume the explicitly stated effects of the action to be the first ones in a causal chain, and that indirect effects are downstream in that chain. For a counterexample, consider the case of a lamp *la* which is turned on independently by two different switches: $la \leftrightarrow sw_1 \vee sw_2$. *la*, sw_1, and sw_2 are the only state variables, and there is no other domain constraint. The action of "turning on the lamp" is defined simply by stating that *la* becomes true. Presumably, ramification shall allow us to conclude that either sw_1 or sw_2 has become true as well, since those are the only ways of turning on the lamp. However, a causal approach will observe that the causal link is from the switches to the lamp, and not vice versa, so it will not deal correctly with this example.

I submit that this is not an odd example: is is often desirable to specify actions in terms of their ultimate and main effects, rather than in terms of the specifics of how to perform them. Reasoning from main effect to instrumental operations, and from there to other effects of the same operations is therefore an important part of the ramification problem.

2.4 APPROACHES ADDRESSED HERE

The present paper is restricted to an analysis of approaches involving minimization, either plain or under categorization of state variables. Our first assessment results for a causation oriented approach have been reported in [21], but they do not fit into the present conference paper. Anyway, the semantics that is proposed and used here is distinctly causation-oriented.

3 UNDERLYING SEMANTICS

The purpose of the underlying semantics is to define the set of intended models, and thereby the set of intended conclusions, in a precise, simple, and intuitively convincing fashion. We shall begin with a naive variant of underlying semantics, and then refine it twice.

3.1 STATE-TRANSITION SEMANTICS

The following basic concepts are used. Let $\mathbf{R}$ be the set of possible states of the world, formed as the Cartesian product of the finite range sets of a finite number of *state variables*. Also, let $\mathbf{E}$ be the set of possible actions, and let the *main next-state function* $\mathbf{N}(E, r)$ be a function from $\mathbf{E} \times \mathbf{R}$ to non-empty subsets of $\mathbf{R}$. The function $\mathbf{N}$ is intended to indicate the set of possible result states if the action E is performed when the world is in state r. The assumption $\mathbf{N}(E, r) \neq \emptyset$ expresses that every action E can always be executed in every starting state r. (The symbol E is derived from "event"; we consider actions and events to be closely related).

The assumption of a finite number of state variables is motivated as long as one can assume a fixed and finite

number of objects *within each scenario*, although different scenarios may involve different object sets. This means that each scenario is essentially propositional in character, even though a first-order syntax is used in the logical formulae.

Now, consider a chronicle (= scenario description) consisting of a specification of the object set, action laws, schedule, and observations, where the action laws specify a next-state function **N** using some appropriate notation, and where time is assumed to be taken as the non-negative integers. Like in previous publications, we understand the "schedule" to specify the actions and their (absolute or relative) time of occurrence, and the "observations" to specify the values of certain state variables (= fluents) at some points in time, before, during, or after the actions. An interpretation for such a chronicle shall be chosen as a history over time, that is, a mapping from timepoints to states. Assignments of values to constant symbols etc. are also needed, but are of marginal importance for the present purpose.

The set of intended models for a chronicle is defined using an *ego-world game*, as follows. First, the observations are removed from the chronicle, so that only the action laws and the schedule remain. The following process is performed using the observation-free chronicle: Start with time as zero and the world in an arbitrary state. At each point in time, perform the move of the "ego" or agent, and the move of the world. The move of the ego is dictated by the schedule, and consists of selecting the action that is indicated by the schedule for the current point in time, or no action if nothing is specified. The move of the world is as follows: if the state of the world at the current time t is r, and the ego has just chosen the action E, then select some r' which is a member of $\mathbf{N}(E, r)$, make it the state of the world at time $t + 1$, and increase the current time by 1. If no move is required, assign the state r to the next timepoint $t + 1$ as well.

The described procedure will non-deterministically generate a set of possible histories of the world. The given definition applies trivially if all actions are specified explicitly by the schedule. Less trivial cases arise if actions are specified in other ways, for example as a disjunction between actions ("either action A or B is performed"), or with an imprecisely specified timing. However, those are technical details, and the general idea remains the same.

Among the world histories that are generated by the ego-world game, one then selects a subset consisting of those histories where all the observations are satisfied. That subset is the set of intended models for the given chronicle. In this case, which is the simplest one of all, the set of intended models will of course be exactly the set of models for the chronicle, using a conventional concept of models. We proceed to less trivial cases, where the characteristic aspects of the frame problem enter the picture.

3.2 PARTIAL-STATE TRANSITION SEMANTICS

The purpose of the partial-state transition semantics is to capture the basic assumption of inertia ("no change except as the explicit result of an action") while retaining the general flavor of the state-transition semantics. We use **R**, **E**, and **N**, like before, but we also introduce an *occlusion function* $\mathbf{X}(E, r)$ which is a function from $\mathbf{E} \times \mathbf{R}$ to subsets of the set of state variables. The occlusion function is intended to specify, for a given action E and starting state r, the set of those state variables whose new value after the action is going to be explicitly stated in the action law.

In the particular case of strict inertia, then, it must be the case that any state variable which is not a member of $\mathbf{X}(E, r)$ has the same value in r and in all the members of $\mathbf{N}(E, r)$. (Thus, it would be correct to select $\mathbf{X}(E, r)$ as a the set of all state variables, but it is intended that smaller sets shall be chosen whenever possible).

The functions **N** and **X** together determine the action laws down to the level of semantic equivalence: if these two functions are given, then the action law for an action E shall be constructed so that for each $r \in \mathbf{R}$, it specifies what $\mathbf{N}(E, r)$ says about the possible new values of the state variables in $\mathbf{X}(E, r)$, and it shall not say anything about the new values of other state variables. (Notice, however, that the new values of state variables in $\mathbf{X}(E, r)$ may equally well *depend on* the values of state variables outside $\mathbf{X}(E, r)$ in the starting state r of the action).

Since we are only interested in the set of new states that are permitted by the action law, and not in its actual syntactic form, it is not necessary to introduce any assumptions about the choice of logical language or about the exact structure of action laws. Using a much simpler approach, we shall introduce a derived function $\mathbf{N}'$, the *stated next-state function* which is obtained from the combination of **N** and **X**, and which is used for defining and analyzing the nonmonotonic entailment methods. This function $\mathbf{N}'$ is obtained in the following manner. For every result state $r' \in \mathbf{N}(E, r)$, identify the set of all states which agree with r' for all state variables in $\mathbf{X}(E, r)$, but which have arbitrary values elsewhere. Construct $\mathbf{N}'(E, r)$ as the union of all those sets. It follows immediately that $\mathbf{N}'(E, r) \supseteq \mathbf{N}(E, r)$.

One might represent $\mathbf{N}'(E, r)$ equivalently as the set of partial states which are restrictions of members of $\mathbf{N}(E, r)$ to the state variables in $\mathbf{X}(E, r)$. This is the reason for the name "partial-state transition semantics".

The function $\mathbf{N}'$ that is obtained in this way is a di-

rect counterpart of the usual, logical formulation of an action law, although we have turned the situation around in one important respect when it comes to ramification later on. Usually, one formulates the action law as a partial description of the effects of the action, and then proceeds to asking "what other effects should one also be able to infer?" Here, instead, we start with the function $\mathbf{N}$ that defines *all* the effects, and then proceed to the function $\mathbf{N}'$ that corresponds to the description of the action.

Notice that $\mathbf{N}'$ is uniquely determined by $\mathbf{N}$ and $\mathbf{X}$. We do not write $\mathbf{X}$ as an index on $\mathbf{N}$ because we are going to need another index below, and double indices are cumbersome. Notice, also, that $\mathbf{N}(E,r) = \emptyset \leftrightarrow \mathbf{N}'(E,r) = \emptyset$. In other words, our assumption of the universal applicability of actions is preserved in the transformations from $\mathbf{N}$ to $\mathbf{N}'$.

4 SEMANTICS-LEVEL VS. LOGIC-LEVEL ANALYSIS

Generally speaking, our task is to analyze logics of actions and change, either by relating them to an underlying semantics, or by relating two such logics to each other. In principle, this should require one to go through the usual routine of defining the syntax of logic formulae, interpretations, etc. In the present case, however, it is possible to do the essential parts of the analysis on the semantic level, and to rely on earlier results for the relation to the conventional parts of the logic.

4.1 RAMIFICATION ON THE SEMANTIC LEVEL

On the semantic level, we view ramification or update as follows. Let $\mathbf{R}_c$ be a subset of $\mathbf{R}$ consisting of those states [called *admitted states*] that satisfy the static domain constraints, and assume from $\mathbf{N}$ that

$$r \in \mathbf{R}_c \rightarrow \mathbf{N}(E,r) \subseteq \mathbf{R}_c. \qquad (1)$$

Given $\mathbf{N}$, $\mathbf{X}$, and $\mathbf{R}_c$, and having derived $\mathbf{N}'$, our task is to obtain from $\mathbf{N}'$ the best possible approximation to $\mathbf{N}$, and ideally to obtain back $\mathbf{N}$ itself.

The first step must be to make full use of $\mathbf{R}_c$, the set of legal world states. Clearly, we only require $\mathbf{N}$ and its approximation to agree when the second argument is a member of $\mathbf{R}_c$, and since we have assumed that $\mathbf{N}(E,r) \subseteq \mathbf{R}_c$ in such cases, we obtain at once

$$r \in \mathbf{R}_c \rightarrow \mathbf{N}(E,r) \subseteq \mathbf{N}'(E,r) \cap \mathbf{R}_c. \qquad (2)$$

If it happens to be the case that equality is obtained here, then no minimization (or other additional measures) are required at all. The use of $\mathbf{N}'(E,r) \cap \mathbf{R}_c$ as an approximation of $\mathbf{N}(E,r)$ will be called *straight ramification*. Then, in logical terms, one can take the action law (characterizing the new values of the state variables occuring in $\mathbf{R}_c$, after the action E has been performed from the starting state r), add the domain constraints, and draw the conclusions using the well-understood methods for strict inertia.

One may think that straight ramification is too simple, and that it will not work in practice. However, it does have the advantage of soundness: it may not draw all intended conclusions, but at least it does not draw any unintended conclusions. In terms of the two rationale for ramification that were mentioned initially, it would seem intuitively that straight ramification should be appropriate for redundancy-motivated ramification, but probably not for propagation-motivated ramification.

4.2 RAMIFICATION ON THE LOGIC LEVEL

When the ramification problem is posed on the level of the concrete logic, one encounters a host of difficulties: the choice of logical language, dealing with the various parts of the scenario description, the relationship between the next-state functions and the corresponding action laws, the problem of non-initial observations which lead to postdiction problems, etc. Fortunately, however, all of these problems have been adequately handled already in the context of strict inertia, and can be considered as routine matters. We refer to [18] for all the necessary details. Therefore, we can proceed entirely on the semantic level here.

5 MEASURE SYSTEMS

We proceed now to the analysis of cases where straight ramification is not sufficient. For our analysis of the commonly used, minimization-based approaches, we introduce the following uniform framework. A *measure system* is a threetuple

$$\langle \mathbf{Q}, d, \prec \rangle, \qquad (3)$$

where $\mathbf{Q}$ is a domain of *quantities*, d is a *difference function* $\mathbf{R} \times \mathbf{R} \rightarrow \mathbf{Q}$, $\prec$ is a strict partial order on $\mathbf{Q}$, and where $q \not\prec d(r,r)$ for all r and q. Notice that even $d(r,r') = d(r',r)$ is not required, which is why we call this a "difference" and not a "distance". We shall say that r' is *less remote* than r'' from r if $d(r,r') \prec d(r,r'')$.

If m is a measure system $\langle \mathbf{Q}, d, \prec \rangle$, then $\mathbf{N}'_m$ shall be the function such that

$$\mathbf{N}'_m(E,r) \subseteq \mathbf{N}'(E,r) \cap \mathbf{R}_c \qquad (4)$$

and where $\mathbf{N}'_m(E,r)$ consists exactly of those members of $\mathbf{N}'(E,r) \cap \mathbf{R}_c$ which are *minimally remote from r according to the measure system m*. Formally, for any member r' of $\mathbf{N}'_m(E,r)$, there must not be any other member r'' of $\mathbf{N}'(E,r) \cap \mathbf{R}_c$ which is less remote than

r' from r, and any $r' \in \mathbf{N}'(E,r) \cap \mathbf{R}_c$ having that property must be a member of $\mathbf{N}'_m(E,r)$.

This concept of measure system makes it straightforward to define a number of current minimization methods. For example, the simple case where one partitions state variables into "occluded" and "unoccluded" ones, and minimizes the set of changes in the unoccluded state variables, is obtained by defining $\mathbf{Q}$ as the power set of the set of state variables, $d(r,r')$ as the set of unoccluded state variables where r and r' assign different values, and $\prec$ as the strict subset relation.

It is easily verified that the approximation $\mathbf{N}'_m$ for arbitrary m satisfies all the Katsuno-Mendelzon postulates [9] **U1** through **U8**, except **U2**. The appropriateness of **U2** has already been cast in doubt by del Val and Shoham in [4], where they observed that **U2** is not well chosen in the case of nondeterministic actions.

6 COMPARATIVE ASSESSMENTS

With the given definitions, a measure system m is *correct* for a main next-state function $\mathbf{N}$ and occlusion function $\mathbf{X}$ iff $\mathbf{N}(E,r) = \mathbf{N}'_m(E,r)$ for every action E and every $r \in \mathbf{R}_c$. The *range of applicability* of a measure system m is the set of main next-state functions $\mathbf{N}$ where m is correct.

If one can not find a correct measure system for a given application, then presumably one prefers a measure system that is *sound* in the sense that it does not allow any unwarranted conclusions. In our framework, the measure system m is sound for a main next-state function $\mathbf{N}$ and occlusion function $\mathbf{X}$ iff $\mathbf{N}(E,r) \subseteq \mathbf{N}'_m(E,r)$ for every action E and every $r \in \mathbf{R}_c$.

It is evident that no measure system is correct for all $\mathbf{N}$, and that straight ramification is the only measure system that is sound for all $\mathbf{N}$. (The latter follows by choosing $\mathbf{N}(E,r)$ so that it equals $\mathbf{N}'(E,r) \cap \mathbf{R}_c$, so that no minimization is desired).

We shall compare different minimization methods with respect to how restrictive they are: a minimization method based on measure system m is *more restrictive* than one based on measure system m' iff $\mathbf{N}'_m(E,r) \subseteq \mathbf{N}'_{m'}(E,r)$ for all E and for all $r \in \mathbf{R}_c$. Thus, a more restrictive method is one which obtains a smaller set of preferred models, and thereby a larger set of conclusions, and a smaller set of pairs $\langle \mathbf{N}, \mathbf{X} \rangle$ for which it is sound. Clearly, straight ramification is the unique least restrictive method.

The following methods are of particular interest.

MSC: Minimal secondary change. State variables are divided into two categories: *occluded* ones specified by $\mathbf{X}$, and *secondary* ones which are the rest. Action laws specify the possible new values for occluded state variables. For given E and r, one selects those members of $\mathbf{N}'(E,r) \cap \mathbf{R}_c$ which minimize the set of secondary state variables that change compared to r. This is the method proposed by del Val and Shoham in [4].

MSCC: Minimal secondary change with changeset-partitioning. Similar to MSC for deterministic actions, but differs for nondeterministic actions. In MSCC, one divides $\mathbf{N}'(E,r) \cap \mathbf{R}_c$ into partitions each consisting of those proposed new states which are equal with respect to occluded state variables. In each of the partitions, one selects the subset of those members which are minimally remote from r, using the same criterion as for MSC. Then one forms the union of those closest subsets. It follows at once that MSCC is less restrictive than MSC, although more restrictive than straight ramification.

The intuition behind this modification is as follows. Consider an action which is nondeterministic already from the point of view of the action law, that is, the action law specifies several alternative outcomes of the action. Then one would like to minimize changes for each of the outcomes separately. For example, the action of playing a game on a gambling machine may result in different states in the machine; let us say just *win* and *lose* for simplicity. Suppose that the action law only specifies this outcome, and there is a static domain constraint specifying that *win* is accompanied by *money-falling-out*. Then MSC will always prefer *lose* over *win* because it minimizes the set of side-effects. MSCC, by comparison, considers *win* and *lose* to be equally preferred. This method was first proposed by Sandewall in [16], page 77.

MRC: Minimal remanent change. State variables are divided into three disjoint categories: occluded, remanent, and dependent ones. The occluded state variables are those whose new values are specified in the action laws, like before. Changes in remanent state variables are minimized, whereas changes in dependent state variables are disregarded for the purpose of minimization. This is essentially the method $\mathcal{AR}_0$ proposed by Kartha and Lifschitz in [8], if one identifies "occluded" with "frame released", "remanent" with "frame unreleased", and "dependent" with "non-frame" state variables. It is also used in PMON(R) proposed by Doherty and Peppas in [5].

MRCC: Minimal remanent change with changeset-partitioning. MRCC uses the same threeway partitioning as in MRC, but minimizes changes separately for each outcome in occluded state variables, like in MSCC.

The intuition behind the three-way partitioning is as follows: even if one chooses to divide the state variables into two categories, one where change is minimized and one where it is not, there may still be two distinct reasons why a state variable is in the latter

category. On one hand, there are those state variables where the outcome or set of possible outcomes is already specified in the action law, so that further minimization is not needed and often not desired (because it may remove some of the intended outcomes of nondeterministic actions). On the other hand, there are those state variables which are functionally dependent on some of the others, and where the dependency is specified by the static domain constraints. In our formulation of the problem, the first subgroup is indicated by $\mathbf{X}$, since $\mathbf{X}$ is used in two ways: it defines $\mathbf{N}'$ from $\mathbf{N}$, and it is part of the definition of minimization. The other subgroup can not be included in $\mathbf{X}$ – the formal account in [20] introduces the required notation.

One might think that since the dependent state variables are functionally determined by the others, it would not hurt to bring them along in the minimization. The problem is, as was shown already by Winslett in the 'Aunt Agatha' scenario, that there may be undesired tradeoffs so that an unintended model containing a change in a remanent state variable may be be accepted because it contains a smaller set of changes in dependent variables. In such scenarios it is necessary to exclude dependent state variables from minimization.

Finally, the list should also include the classical method:

MC: Minimization of change. No partitioning of the state variables. One selects the subset of $\mathbf{N}'(E,r) \cap \mathbf{R}_c$ consisting of those members minimizing the set of state variables where the value differs from r. This is the method originally proposed by Winslett [24].

It is straightforward to define these methods in terms of measure systems. The definitions can be obtained from [20].

Writing $m \leq m'$ when m is more restrictive than m', and **SR** for straight ramification, we have proved the following inclusion results:

Theorem 1

$$\mathbf{MC} \leq \mathbf{MSCC} \leq \mathbf{SR} \tag{5}$$

$$\mathbf{MSC} \leq \mathbf{MSCC} \leq \mathbf{SR} \tag{6}$$

$$\mathbf{MRC} \leq \mathbf{MRCC} \leq \mathbf{SR} \tag{7}$$

The proofs are found in our technical report [20]. There does not seem to be any similar result relating e.g. MSCC and MRCC.

These results suggest at once that unless there are specific reasons to the contrary, one ought to prefer methods with changeset-partitioning over methods without, that is, MSCC over MSC and MRCC over MRC. This is because one will then obtain a method with a larger range of sound applicability. In other words, although possibly one will fail to obtain some intended conclusions, at least there is a smaller possibility that the system actually infers non-intended conclusions.

The methods with changeset-partitioning also have another advantage, namely that they are monotone with respect to the set of occluded features, $\mathbf{X}$. The precise result is as follows.

Theorem 2 *If* MSCC *is incorrect for a certain choice of* $\mathbf{N}$ *and* $\mathbf{X}$, *and* $\mathbf{X}'(E,r) \subseteq \mathbf{X}(E,r)$ *for every choice of* E *and* r, *then* MSCC *is incorrect for* $\mathbf{N}$ *and* $\mathbf{X}'$ *as well.*

The proof is found in [20]. The same theorem holds for MRCC as long as the set of dependent state variables is held fixed.

Therefore, it is meaningful to ask for a *smallest* choice of action laws, in the spirit of the initial quotation from Ginsberg and Smith: a choice of action laws that minimizes the set of state variables where the laws specify the new values after the action has been performed, and still the inference mechanism obtains the intended conclusions. The corresponding methods without changeset-partitioning do not have that monotonicity property.

7 CAUSAL PROPAGATION SEMANTICS

We now proceed to the underlying semantics which is the necessary prerequisite for the analysis of range of (sound) applicability. The purpose of the underlying semantics is to define the set of intended models, and thereby the set of intended conclusions, in a precise, simple, and intuitively convincing fashion. In the case of underlying semantics for ramification, we consider it particularly important to capture the propagation-oriented reason for ramification. We propose to use the following *causal propagation semantics*, which is obtained through a second modification of the basic state transition semantics described above.

The domain $\mathbf{R}$ of world states and the domain $\mathbf{E}$ of actions are defined like before. A binary non-reflexive *causal transition relation* C between states is introduced; if $C(r,r')$ then r' is said to be the *successor* of r. A state r is called *stable* iff it does not have any successor. The set $\mathbf{R}_c$ of *admitted states* is chosen as a subset of $\mathbf{R}$ all of whose members are stable with respect to C. (It may or may not be chosen to contain all the stable states).

Furthermore, we assume an *action invocation relation* $G(E,r,r')$ where $E \in \mathbf{E}$ is an action, r is the state where the action E is invoked, and r' is the new state where the instrumental part of the action has been executed. For every E and r there must be at least one r' such that $G(E,r,r')$, that is, every action is

always invokable.

An *action system* is a tuple $\langle \mathbf{R}, \mathbf{E}, C, \mathbf{R}_c, G \rangle$, and is intended as follows. For any state $r \in \mathbf{R}_c$, consider a state r_1 such that $G(E, r, r_1)$, and a sequence of states

$$r_1, r_2, \ldots r_k$$

where $C(r_i, r_{i+1})$ for every i between 1 and $k-1$, and r_k is a stable state. Such a sequence is called a *transition chain*, and r_k is considered as a *result state* of the action E in the sitution r.

We intend, of course, that it shall be possible to characterize the result states r_k in terms of E and r, but without referring explicitly to the details of the intermediate states. The following assumptions (or something similar) are needed in order to make this work as intended.

Definition 1 *If three states r, r_i, and r_{i+1} are given, we say that the pair r_i, r_{i+1}* **respects** *r iff*

$$r_i(f) \neq r_{i+1}(f) \rightarrow r_i(f) = r(f),$$

for any state variable f that is defined in $\mathbf{R}$. Then, an action system $\langle \mathbf{R}, \mathbf{E}, C, \mathbf{R}_c, G \rangle$ is said to be **respectful** *iff, for every $r \in \mathbf{R}_c$, every $E \in \mathbf{E}$, r is respected by every pair r_i, r_{i+1} in every transition chain, and the last element of the chain is a member of $\mathbf{R}_c$.* □

This condition amounts to a "write-once" or "single-assignment" property: if the action E is performed from state r, the world may go through a sequence of states, but in each step from one state to the next, there cannot be changes in state variables which have already changed previously in the sequence, nor can there be any additional change in a state variable that changed in the invocation transition from r to r_1. This condition is also analogous to Lin's definition of a stratified system in [11].

As a consequence of these definitions, we obtain that if $\langle \mathbf{R}, \mathbf{E}, C, \mathbf{R}_c, G \rangle$ is a respectful action system, if $r \in \mathbf{R}_c$, $E \in \mathbf{E}$, and $G(E, r, r')$ holds for some r', then all the transition chains that emerge from (E, r) are finite and cycle-free. The set of result states will be denoted $\mathbf{N}(E, r)$, using the same notation as in the state-transition semantics. Thus, the main next-state function $\mathbf{N}(E, r)$ which was previously taken as given, is here taken as derived from the elementary relations G and C in the action system. Also, $\mathbf{N}(E, r) \neq \emptyset$.

Respectful action systems are intended to capture the basic intuitions of actions with indirect effects which are due to causation, as follows. Suppose the world is in a stable state r, and an action E is invoked. The immediate effect of this is to set the world in a new state, which is not necessarily stable. If it is not, then one allows the world to go through the necessary sequence of state transitions, until it reaches a stable state. *That whole sequence* of state transitions is together viewed as the action, and the resulting admitted state is viewed as the result state of the action.

The assumption of sequentiality of actions within a chronicle is then taken to mean that the agent or "ego" abstains from invoking any new action before the previous action has reached a stable state. The exact interpretation of the state transitions may vary: as spontaneous transitions in the physical world, or as more or less logical conclusions about a world where all changes happen (or are modelled as happening) concurrently. One may even include the case where additional interventions by the agent (viewed as a controller) are necessary along the path from the starting state to the result state of the action.

For example, in the lamp scenario that was mentioned in section 2.3, if the state variables represent the first switch, the second switch, and the lamp, respectively, the invocation relation G will contain argument sequences such as $(E_1, \mathbf{FFF}, \mathbf{TFF})$ for turning on the first switch; the causal transition relation will contain the argument sequence $(\mathbf{TFF}, \mathbf{TFT})$ expressing that if the first switch is on, then the lamp goes on as well. The action E_2 which is instrumentally defined as turning on either of the two switches is true for both $G(E_2, \mathbf{FFF}, \mathbf{TFF})$ and $G(E_2, \mathbf{FFF}, \mathbf{FTF})$.

The idea with ramification, now, is that one does not wish to specify *all* the effects of an action in the action law; one wishes to restrict the action law to the most important effects, and other effects should be deducible if needed. In other words, the action law shall not specify all components of the result state, and not even all the components that change, but only some of them. Other changes are deduced using information about the relations C and G.

Causal propagation semantics is not applicable to all kinds of ramification. For example, it would be completely unnatural in the context of physical movement, where the displacement of one object may cause another object to move as well. We are currently working on other semantics for systems involving coupled continuous change.

Notice that it is not necessary to let the action law characterize the invocation relation G. Alternatively, the action law may characterize some of the later changes along the chain, namely, if the indirect changes are considered to be the most important ones in terms of describing the action. For example, the action law for E_2 in the second-previous paragraph may specify the change in the state variable indicating that "the lamp is lit", leaving the "upstream" change of one or the other of the switches to be inferred. In general, the non-explicit effects of the action may be a combination of changes that are causationally "downstream" as well as "upstream" of the explicit ones.

8 SOUNDNESS ASSESSMENTS FOR MSCC

After having related the various minimization methods to each other, we proceed now to a first analysis of how they relate to the underlying semantics.

The general problem is viewed as follows. Let the following be given:

- A respectful action system $\langle \mathbf{R}, \mathbf{E}, C, \mathbf{R}_c, G \rangle$. Its corresponding main next-state function will be written $\mathbf{N}$, as before.
- An occlusion function, $\mathbf{X}$.
- A measure system $m = \langle \mathbf{Q}, d, \prec \rangle$.

Informally, it is still the case that $\mathbf{X}$ indicates what are the state variables whose new values are indicated in the action law. The functions $\mathbf{N}$ and $\mathbf{X}$ together define the stated next-state function $\mathbf{N}'$, as described above. The measure system is said to be *correct* for the action system and the occlusion function iff

$$\mathbf{N}'_m(E, r) = \mathbf{N}(E, r)$$

for every E and r. Likewise, the measure system is said to be *sound* for the action system and the occlusion function iff

$$\mathbf{N}'_m(E, r) \supseteq \mathbf{N}(E, r)$$

for every E and r. Notice how the entities in this equation have been derived: $\mathbf{N}$ from the action system alone, $\mathbf{N}'$ from $\mathbf{N}$ and $\mathbf{X}$, and $\mathbf{N}'_m$ from $\mathbf{N}'$ and m.

In particular, given one of the specific measure systems and entailment methods that were described above, for example MRCC, a particular partitioning of the state variables is said to be *correct* for the action system, the occlusion function, and the method at hand iff the corresponding measure system is correct for them. Soundness is defined similarly.

The corresponding *assessment problem* for an entailment method is defined as the following question: what are the requirements on the combination of a respectful action system, an occlusion function, and a partitioning whereby the given entailment method is correct (or sound)?

8.1 OCCLUSION OF INVOCATION-CHANGED STATE VARIABLES

We consider separately the case where action laws describe invocations, and the case where they describe downstream effects.

Definition 2
A respectful action system $\langle \mathbf{R}, \mathbf{E}, C, \mathbf{R}_c, G \rangle$ *is said to have* **fixed control** *iff there is some subset* $\mathbf{F}_c$ *of the set* $\mathbf{F}$ *of state variables, such that*

$$G(E, r, r') \rightarrow dom(r - r') \subseteq \mathbf{F}_c$$

and

$$C(r, r') \rightarrow dom(r - r') \cap \mathbf{F}_c = \emptyset$$

The set $\mathbf{F}_c$ *is called the* control set. □

Here and henceforth the following conventions are used: $-$ denotes set subtraction, mappings are viewed as sets of argument/value pairs, and $dom(r)$ maps a mapping (in this case, a partial state) to its domain.

Members of the control set will be called *controlled* state variables; the others will be called *obtained* state variables.

This definition means that invocation transitions can only influence state variables in the control set, and causal transitions can never influence state variables in the control set. Thielscher's relay example in [22] is an example of a system which does not have fixed control.

Definition 3 *Given a state domain* $\mathbf{R}$, *a state* r'' *is said to be a* **hybrid** *between two states* r *and* r' with respect to *a set* $F \subseteq \mathbf{F}$ *of state variables iff*

(1) *for every state variable* f,

$$r''(f) = r(f) \vee r''(f) = r'(f),$$

(2) *if* $f \in F$ *then* $r'(f) = r''(f)$, *and*

(3) r'' *is different from both* r *and* r'. □

Definition 4 *A respectful action system is said to be* **hybrid free** *with respect to a set* F *of state variables iff no two members* r *and* r' *of* $\mathbf{R}_c$ *have a hybrid with respect to* F *which is also a member of* $\mathbf{R}_c$. □

It follows that if an action system is hybrid free with respect to a set F, then it is hybrid free with respect to any superset of F. (Any hybrid with respect to the superset would be a hybrid with respect to F). We immediately obtain:

Theorem 3 **MSCC** *is sound for the combination of a respectful action system* $\langle \mathbf{R}, \mathbf{E}, C, \mathbf{R}_c, G \rangle$ *and an occlusion function* $\mathbf{X}$ *if the following conditions are satisfied:*

1. *The action system has fixed control with a control set* $\mathbf{F}_c$
2. *The action system is hybrid free with respect to* $\mathbf{F}_c$
3. *If* $G(E, r, r')$ *then* $dom(r - r') \subseteq \mathbf{X}(E, r)$

Proof. We are to prove that $\mathbf{N}'_{\mathrm{MSCC}}(E, r) \supseteq \mathbf{N}(E, r)$. Suppose the opposite is the case, that is, for some E and r there is some $r^* \in \mathbf{N}(E, r)$ which is not a member of $\mathbf{N}'_{\mathrm{MSCC}}(E, r)$. Since it is a member of $\mathbf{N}(E, r)$, r^* must be a member of $\mathbf{N}'(E, r)$ and of $\mathbf{R}_c$, so its non-membership of $\mathbf{N}'_{\mathrm{MSCC}}(E, r)$ must be due to the existence of some $r' \in \mathbf{N}'(E, r) \cap \mathbf{R}_c$ which is less remote than r^* from r.

By the definition of MSCC, r' must agree with r^* in all occluded state variables. By assumptions 1 and 3, r^* agrees with r in all non-occluded controlled state variables, so r' must do so as well. Therefore, r^* and r' agree in all state variables in $\mathbf{F}_c$. But r' differs from r in a subset of the state variables where r^* differs from r. Therefore, either r' is a hybrid between r and r^* with respect to $\mathbf{F}_c$, or $r' = r$. The former case is inconsistent with assumption 2 given that $r' \in \mathbf{R}_c$. The latter, bizarre case requires that r and r^* are equal in all controlled state variables, which means that $\mathbf{N}(E, r) = \{r\}$ and $r^* = r$, making it impossible for any r' to be less remote from r than r^* is. □

8.2 OCCLUSION OF NON-INVOKED STATE VARIABLES

We proceed now to the case where action laws characterize other changes of state variables than those involved in the invocation.

Definition 5 *A respectful action system* $\langle \mathbf{R}, \mathbf{E}, C, \mathbf{R}_c, G \rangle$ *is* **unary controlled** *iff* $G(E', r, r')$ *implies that* $dom(r - r')$ *is a singleton, for any* E, r, *and* r'. □

Note that this property is realistic for low-level actions in engineered systems, where controlled state variables represent the actuators.

Definition 6 *Consider the combination of a respectful action system* $\langle \mathbf{R}, \mathbf{E}, C, \mathbf{R}_c, G \rangle$ *having fixed control, and an occlusion function* $\mathbf{X}$. *A state* $r' \in \mathbf{R}_c$ *is called a* **shortcut** *for* (E, r) *in this combination, where* $r \in \mathbf{R}_c$, *if and only if there is some* $r'' \in \mathbf{N}(E, r)$ *such that*

(1) *for every state variable* f,

$$r'(f) = r(f) \vee r'(f) = r''(f),$$

(2) r' *equals* r *in all controlled state variables, and*

(3) r' *equals* r'' *for all state variables in* $\mathbf{X}(E, r)$. □

Example 7 *The following is an example of a shortcut. Consider an electric circuit consisting of a number of switches, lamps, and relays, where the switches and only them are controlled. In the current state* r *of the circuit, changing a particular switch* s *will turn on a lamp* l *which is presently off; there may also be other effects. Let* r'' *be the resulting state. An action* E *is defined so that* $\mathbf{X}(E, r) = \{l\}$ *and* r'' *is a member of* $\mathbf{N}(E, r)$, *that is, the action is defined in terms of the effect of turning on the lamp, and not in terms of changing the switch. However, there is also a sequence of other actions where different switches are changed in succession, and which ends up in a state* r' *where all switches are in the same position as in the present state* r, *the lamp* l *is on anyway, and only a subset of the changes from* r *to* r'' *occur in* r'. □

It would seem that the property of having a shortcut is not a particularly natural one for an action system to have. However, there is an important special case, namely for the combination of an action system and an occlusion function where an action possibly has a null stated effect, that is, there is some $r'' \in N(E, r)$ for which r and r'' are equal for all occluded state variables. In this case, r itself is a shortcut for (E, r).

We immediately obtain:

Theorem 4 **MSCC** *is sound for the combination of a respectful action system* $\langle \mathbf{R}, \mathbf{E}, C, \mathbf{R}_c, G \rangle$ *and an occlusion function* $\mathbf{X}$ *if the following conditions are satisfied:*

1. *The action system has fixed control with a control set* $\mathbf{F}_c$.

2. *The action system is hybrid free with respect to* $\mathbf{F}_c$.

3. $\mathbf{X}(E, r)$ *is disjoint from* $\mathbf{F}_c$ *for every* E *and* $r \in \mathbf{R}_c$.

4. *The action system is unary controlled.*

5. *The combination of the action system and the occlusion function does not have any shortcut.*

Proof. We are to prove that $\mathbf{N}'_{\mathrm{MSCC}}(E, r) \supseteq \mathbf{N}(E, r)$. Suppose the opposite is the case, that is, for some E and r there is some $r^* \in \mathbf{N}(E, r)$ which is not a member of $\mathbf{N}'_{\mathrm{MSCC}}(E, r)$. Since it is a member of $\mathbf{N}(E, r)$, r^* must be a member of $\mathbf{N}'(E, r)$ and of $\mathbf{R}_c$, so its non-membership of $\mathbf{N}'_{\mathrm{MSCC}}(E, r)$ must be due to the existence of some $r' \in \mathbf{N}'(E, r) \cap \mathbf{R}_c$ which is less remote than r^* from r.

Consider the causation chain leading up to r^*, and its first element r_1. Because of the unary control property, r and r_1 will differ in exactly one state variable; this state variable is a controlled one because of assumption 1, and it is not occluded because of assumption 3. There are two possibilities with respect to r':

(1) r' equals r_1 in all controlled state variables. Then r' is a hybrid between r and r^*, violating assumption 2.

(2) r' equals r in all controlled state variables. Since r' is a member of $\mathbf{N}'(E, r)$, it must agree with some member r'' of $\mathbf{N}(E, r)$ in all occluded state variables. Therefore, r' is a shortcut for (E, r), which violates assumption 5. □

The results of these theorems do not apply in general for MSC because of interference between primary outcomes of the action. Generalization of these results to MRCC encounters technical complications in the generalization of the "hybrid" concept.

These assessments may be compared with the previous results on assessment of entailment methods with respect to the assumption of strict inertia. Those assessments [18] made use of the concept of minimal-

change-compatible, which in the present context may be equivalently defined as follows: An action system is **minimal-change-compatible** iff it is not the case, for any r, E, r', and r'', that $r' \in \mathbf{N}(E, r)$, $r'' \in \mathbf{N}(E, r)$, and r'' is a hybrid between r and r' with respect to $\emptyset$. We showed in [18] that PCMF (prototypical minimization of change with filtering) is correct for all minimal-change-compatible chronicles with single-timestep actions. We observe that if an action system is hybrid free with respect to $\emptyset$, then it is both minimal-change-compatible and hybrid free with respect to an arbitrary set F.

8.3 EXAMPLES

The soundness property means that the entailment method MSCC does not miss any intended models, but it still allows for the possibility of including some unintended models. The following two examples illustrate how unintended models may be included in $\mathbf{N}'_{\mathrm{MSCC}}(E, r)$.

Example 8 *For the first example, suppose* $\mathbf{R}$ *is selected as* $\{\mathtt{T},\mathtt{F}\} \times \{1,2\} \times \{1,2\}$, *where the first component is controlled and always occluded, and the other two are secondary. Suppose further that the causal transition function* C *is true for the following pairs:*

$$C(\mathtt{T11},\mathtt{T12}),\quad C(\mathtt{T22},\mathtt{T21}),$$
$$C(\mathtt{F12},\mathtt{F11}),\quad C(\mathtt{F21},\mathtt{F22})$$

and that $\mathtt{T12},\mathtt{T21},\mathtt{F11},\mathtt{F22}$ *are in* $\mathbf{R}_c$. *The invocation relation* G *is supposed to contain*

$$G(E_1,\mathtt{F11},\mathtt{T11}),\quad G(E_1,\mathtt{F22},\mathtt{T22})$$

as well as some suitable transitions when the controlled state variable is initially T. *It is clear that* $\mathbf{N}(E_1,\mathtt{F11}) = \{\mathtt{T12}\}$ *and* $\mathbf{N}'_{\mathrm{MSCC}}(E_1,\mathtt{F11}) = \{\mathtt{T12},\mathtt{T21}\}$, *so soundness but not correctness is obtained.* □

The problem here, of course, is that the entailment method MRCC is based on a notion of minimal change which does not use any of the causal transition information from C.

Example 9 *The following example illustrates another possibility for the introduction of unintended models. Suppose now that* $\mathbf{R}$ *is selected as* $\{\mathtt{T},\mathtt{F}\} \times \{\mathtt{T},\mathtt{F}\} \times \{1,2\}$, *where the first two components are controlled and the third one is obtained. Suppose further that the transition function* C *satisfies*

$$C(\mathtt{TF1},\mathtt{TF2}),$$

and that $\mathtt{FF1},\mathtt{FT1},\mathtt{TF2},\mathtt{TT1}$ *are in* $\mathbf{R}_c$. *The invocation relation* G *is supposed to contain*

$$G(E_1,\mathtt{FF1},\mathtt{TF1}),\quad G(E_1,\mathtt{FT1},\mathtt{TT1})$$

as well as some suitable invocation transitions for the other members of $\mathbf{R}_c$. *Then* $\mathbf{N}(E_1,\mathtt{FF1}) = \{\mathtt{TF2}\}$. *In this case* $\mathbf{X}(E_1,\mathtt{FF1})$ *needs only contain the first state variable, so both the second and the third state variable can be selected as secondary. Using* MSCC *with that assumption one obtains* $\mathbf{N}'_m(E_1,\mathtt{FF1}) = \{\mathtt{TF2},\mathtt{TT1}\}$, *so again soundness but not correctness is obtained.* □

In this case, the unintended model entered because $\mathbf{X}(E, r)$ is only required to be a subset of the controlled state variables. This opens the door for admitting some model which satisfies the action laws with respect to occluded variables, but which also includes some change in a controlled, non-occluded variable (in this case, the second state variable), provided that it thereby obtains a smaller set of changes in remanent state variables.

The source of incompleteness that was illustrated here can be removed by requiring that all controlled state variables must be occluded, that is, their new value must be stated in the action law. This means effectively that the action law must not merely specify the new values for some state variables that change their value, but also specify explicitly and un-overridably that some state variables do not change their value in the current action. An additional possibility is to introduce a fourth category for those state variables which *must not* change as the result of an action.

9 SUMMARY AND CONCLUSIONS

We published our main results on underlying semantics and assessment of range of applicability for strict inertia in 1993-94. It has been surprisingly difficult to find a natural extension to the case of ramification, and the causal propagation semantics proposed here is the first real advancement. At the same time, we observe that only soundness assessments have been obtained so far, and we think it is likely that soundness assessments will continue to predominate. Therefore, there will be a continuing need for comparative assessments, providing information on the relative size of the selected model sets for different entailment methods.

The main concrete results in the present article are therefore:

- The causal propagation semantics
- The comparative assessments of a number of entailment methods
- The soundness assessment of one of the entailment methods, MSCC
- The observation of the usefulness of changeset-partitioning in minimization-based approaches.

Acknowledgements

The first part of this research was performed while the author was a visiting researcher at LAAS-CNRS,

Toulouse, France. We acknowledge valuable comments on that part of the work by Marie-Odile Cordier and Pavlos Peppas.

References

[1] Gerhard Brewka and Joachim Hertzberg. How to do things with worlds: On formalizing actions and plans. *Journal of Logic and Computation*, 1993.

[2] Marie-Odile Cordier and Pierre Siégel. A temporal revision model for reasoning about world change. In *International Conference on Knowledge Representation and Reasoning*, pages 732–739, 1992.

[3] Marie-Odile Cordier and Pierre Siégel. Prioritized transitions for updates. In C. Froidevaux and J. Kohlas, editors, *Proc. European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 142–151. Springer Verlag, 1995.

[4] Alvaro del Val and Yoav Shoham. Deriving properties of belief update from theories of actions (II). In *International Joint Conference on Artificial Intelligence*, pages 732–737, 1993.

[5] Patrick Doherty and Pavlos Peppas. A comparison between two approaches to ramification: PMON(R) and $\mathcal{AR}_0$. In *Proc. Australian AI Conference*, 1995.

[6] Matthew L. Ginsberg and David E. Smith. Reasoning about actions I: A possible worlds approach. In Frank Brown, editor, *Proceedings of the 1987 Workshop on the Frame Problem in Artificial Intelligence*, pages 233–258. Morgan Kaufmann, Lawrence, Kansas, 1987.

[7] Joakim Gustafsson and Patrick Doherty. Embracing occlusion in specifying the indirect effects of actions. In *International Conference on Knowledge Representation and Reasoning*, 1996.

[8] G. Neelakantan Kartha and Vladimir Lifschitz. Actions with indirect effects (preliminary report). In *International Conference on Knowledge Representation and Reasoning*, pages 341–350, 1994.

[9] Hirofumi Katsuno and Alberto O. Mendelzon. A unified view of propositional knowledge base update. In *International Joint Conference on Artificial Intelligence*, pages 1413–1419, 1989.

[10] Vladimir Lifschitz. Frames in the space of situations. *Artificial Intelligence*, 46:365–376, 1990.

[11] Fangzhen Lin. Embracing causality in specifying the indirect effects of actions. In *International Joint Conference on Artificial Intelligence*, 1995.

[12] Norman McCain and Hudson Turner. A causal theory of ramifications and qualifications. In *International Joint Conference on Artificial Intelligence*, 1995.

[13] Tommy Persson and Lennart Staflin. A causation theory for a logic of continuous change. In *European Conference on Artificial Intelligence*, pages 497–502, 1990.

[14] Erik Sandewall. Non-monotonic entailment for reasoning about time and action, part I: Sequential actions. Technical report, IDA, 1988.

[15] 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.

[16] Erik Sandewall. Features and fluents. Review version of 1991. Technical Report LiTH-IDA-R-91-29, IDA, 1991.

[17] Erik Sandewall. The range of applicability of nonmonotonic logics for the inertia problem. In *International Joint Conference on Artificial Intelligence*, 1993.

[18] Erik Sandewall. *Features and Fluents. The Representation of Knowledge about Dynamical Systems. Volume I.* Oxford University Press, 1994.

[19] Erik Sandewall. Reasoning about actions and change with ramification. In J. van Leeuwen, editor, *Computer Science Today*. Springer Verlag, Berlin. Lecture Notes in Computer Science, Anniversary Volume 1000, 1995.

[20] Erik Sandewall. Systematic comparison of approaches to ramification using restricted minimization of change. Technical Report LiTH-IDA-R-95-15, IDA, 1995.

[21] Erik Sandewall. Transition cascade semantics and first assessments results for ramification. In Oliviero Stock, editor, *Spatial and Temporal Reasoning*. Kluwer Publishing Company, 1997.

[22] Michael Thielscher. An analysis of systematic approaches to reasoning about actions and change. In P. Jorrand, editor, *International Conference on Artificial Intelligence: Methodology, Systems, Applications (AIMSA)*, Sofia, Bulgaria, 1994. World Scientific Publishing Co.

[23] Michael Thielscher. Computing ramifications by postprocessing. In *International Joint Conference on Artificial Intelligence*, 1995.

[24] Marianne Winslett. Reasoning about actions using a possible models approach. In *National (US) Conference on Artificial Intelligence*, pages 89–93, 1988.

[25] Choong-Ho Yi. Towards the assessment of logics for concurrent actions. In Dov M. Gabbay and Hans Jürgen Olbach, editors, *Conference on Formal and Applied Practical Reasoning, FAPR 96*, pages 679–690. Springer Verlag, 1996.

Planning

Strategic Advice for Hierarchical Planners

Karen L. Myers
Artificial Intelligence Center
SRI International
333 Ravenswood Ave.
Menlo Park, CA 94025
myers@ai.sri.com

Abstract

AI planning systems have traditionally operated as stand-alone blackboxes, taking a description of a domain and a set of goals, and automatically synthesizing a plan for achieving those goals. Such designs severely restrict the influence that users can have on the resultant plans. This paper describes an Advisable Planner framework that marries an advice-taking interface to AI planning technology. The framework is designed to enable users to interact with planning systems at high levels of abstraction in order to influence the plan generation process in terms that are meaningful to them. Advice consists of task-specific constraints on both the desired solution and the refinement decisions that underlie the planning process. The paper emphasizes *strategic advice*, which expresses recommendations on how goals and actions are to be accomplished. The main contributions are a formal language and semantics for strategic advice, and a sound and complete HTN-style algorithm for generating plans that satisfy advice.

1 Introduction

Artificial Intelligence (AI) planning technology provides powerful tools for solving problems that involve the coordination of actions in the pursuit of specified goals. The AI community has produced several planning systems whose demonstrations on realistic problems attest to the value of automated planning techniques. Nevertheless, there has been limited success in transitioning this technology to user communities. A major reason for the lack of technology transfer lies with the difficulty of using planning systems. AI planners have traditionally been designed to operate as 'blackboxes': they take a description of a domain and a set of goals and automatically synthesize a plan for achieving the goals. This design has several drawbacks. First, it explicitly limits the amount of influence that a user can have on the generated plans. Second, it requires complete and accurate formalizations of the domain, since the system is expected to operate without user intervention. Providing such comprehensive domain information is time-consuming and expensive, and represents a significant investment for each new application.

Recent trends toward *mixed-initiative* styles of planning have led to support for certain low-level interactions on the part of humans, such as ordering goals for expansion, selecting operators to apply, and choosing instantiations for planning variables [3, 16]. While a step in the right direction, these interactions are too fine-grained for most users, who want to be involved with the planning process at a higher, more strategic level.

This paper describes an effort to make AI planning technology more accessible and controllable via the paradigm of *advice-taking*. An *advisable planner* (AP) will accept a variety of instructions and advice from a user and employ those directives to guide plan construction. Advice can encompass a broad range of constructs, including partial sketches of plans for achieving a set of goals, specific subgoals to be used in pursuit of the overall objectives, and proscriptions and prescriptions on the use of specific objects and actions in certain contexts. Such advice will enable users to interact with a planning system at high levels of abstraction in order to guide and influence the planning process, with the planning system performing the time-consuming work of filling in necessary low-level details. As such, the AP model of planning embodies a shift in perspective on how planning systems should be designed: an Advisable Planner is a tool for enhancing the skills of domain users, not a replacement for them.

Within our model, advice differs from the forms of knowledge traditionally used to represent a planning domain (operator schemas, world models, *etc.*). Rather than expressing general properties of the do-

main, advice encodes *session-specific* recommendations on the application of traditional planning knowledge. As such, advice is an adjunct to the underlying domain knowledge rather than an extension to it. As will be seen, advice serves as a filter on the set of solutions to a given planning problem, thus enabling customizations to suit the needs of individual users.

To illustrate the utility of advice, we present examples from a travel planning domain. A typical HTN planner would allow a user to sketch a high-level outline of a trip, specifying information such as which locations to visit at what times and for what overall cost. The planner would then fill in the appropriate details. Most individuals, however, want to influence their itineraries to a much greater extent, specifying details such as the modes of transportation for various legs, individual carriers to use, accommodation requirements, and restrictions on costs for various aspects of the trip. Our theory of advice enables user customization of generated plans in this way.

This paper lays the foundations for the Advisable Planner. Here we emphasize *strategic advice*, which expresses recommendations on how goals and actions are to be accomplished. Section 2 presents an overall framework for an advisable planner. Section 3 describes strategic advice and its uses. Section 4 presents the formal model of HTN planning that underlies our theory of advice. Section 5 defines a formal language for representing strategic advice, while Section 6 defines *satisfaction* for advice. Section 7 defines a sound and complete HTN-style algorithm for generating plans that satisfy strategic advice. Section 8 discusses related work.

2 The Advisable Planner

This section outlines our vision for an *advisable planner* (AP), describing both the overall architecture and the principal classes of advice that an AP should support. As such, it establishes a conceptual framework in which to ground the more technical work that follows.

2.1 AP Framework

The Advisable Planner model consists of an advice-taking interface layered on top of a core planning system. Advice-taking augments the capabilities of the underlying planning system in the sense that the system does not require advice for its operation. Rather, advice simply influences the set of solutions that the system will provide for a given task. Overall, the AP contains two distinct phases: the *advice translation* phase, and the *problem-solving* phase.

Advice translation involves mapping from user-supplied advice into appropriate internal representations for the planner. The translation process involves several stages (see Figure 1). User advice, specified in some natural (or pseudo-natural) language, is *parsed* into an intermediate normalized representation. The normal form provides a planner-independent representation of the advice, thus enabling a clean semantic definition and portability amongst different planners. *Advice compilation* is the planner-dependent translation from normalized advice to internal constraints defined in terms of planner-specific operators, goals, and individuals. These constraints will be used to direct the plan construction process.

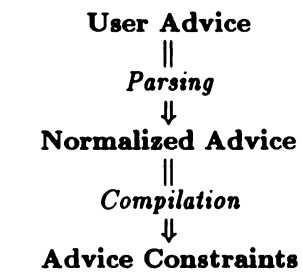


Figure 1: Phases of Advice Representation and Translation

The problem-solving phase takes the compiled advice representations, and generates only solutions that satisfy the advice (referred to as *advice enforcement*). Planning proceeds in a mixed-initiative style whereby the user can make planning-time requests to modify current plans or previously stated advice. The AP may request additional domain information from the user during planning to aid in resolving detected tradeoffs in the plan, to recover from planning failures, or to clarify user-supplied advice.

This paper focuses on the presentation of a basic theory of strategic advice and accompanying algorithms for advice enforcement, ignoring the many complex issues related to advice parsing. In particular, we assume the provision of normalized advice in a language shared with the underlying planning system. For exposition reasons though, we generally use natural-language style presentations of advice.

2.2 Advice Categories

Many kinds of advice can be fruitfully employed for generative planning. *Task advice* designates specific goals to be achieved and actions to be performed, thus amounting to partial specification of a solution to a planning task. A plan sketch constitutes a form of task advice, as do directives such as Do Task-1 before Task-2 and Include Task-5 in the plan. *Evaluational advice* encompasses constraints on some metric defined for the overall plan, such as resource usage, execution time, or solution quality. The directive Spend less than $800 on all accommodations is an example.

This paper focuses on *strategic advice*, which expresses constraints on how to solve tasks, in terms of both specific approaches to pursue and entities to employ. Strategic advice comes in two forms: *role* and *method*. Role advice constrains the use of domain entities in solving tasks, while method advice constrains the type of approach used. As will be seen, strategic advice designates not only properties of the resultant partially ordered set of actions that is generally viewed as 'the plan', but also the underlying justifications for that solution. For this reason, advice enforcement requires explicit representation of those justification structures.

Our model of advice does not encompass *control* of problem-solving [2, 14], such as the order in which to expand goals or instantiate variables. While control is an important issue for automated planning systems, effective control of the planning process requires deep insight into the mechanics of the planning system itself. As such, it is not a responsibility that should be borne by the user (who generally will not be an expert in AI planning) and thus is not an appropriate topic for user advice.

3 Strategic Advice

Strategic advice is formulated in terms of prescriptions and restrictions on *roles*, *fillers*, and *activities*. *Activities* represent abstract operations relative to the underlying planning domain, and are defined in terms of *features* and *roles*. A feature designates a characteristic of interest for an activity. For travel planning, there may be transport activities, vacation activities, bike activities, and accommodation activities; each of these characteristics could be modeled as a feature. A given activity can have multiple features; for example, an activity corresponding to a bike tour could have the features Vacation, Bike, and Inexpensive. *Roles* correspond to capacities in which domain individuals are to be used in an activity. For instance, transport activities could have roles such as Origin, Destination, and Carrier. *Fillers* are specifications of objects to be used in certain roles; they may name explicit individuals, or consist simply of a set of required and prohibited attributes.

As described later, activities, roles, and fillers are grounded in planning constructs such as goals, operators, variables, and bindings. The benefit of the activities/roles framework is that it provides users with a simpler, more abstract model for expressing advice than do the low-level planning constructs.

3.1 Role Advice

Role advice either prescribes or restricts the use of domain entities for filling certain capacities in the plan. Role advice is characterized by the template: `<Use/Don't Use> <object> in <role> for <target-activity>`.

In general, role advice consists of one or more object-role specifications (called a *role-fill*), a *target activity*, and a *polarity* indicating whether the advice is prescribing or prohibiting the role-fill. The following directives provide examples of role advice:

> Stay in 3-star ensuite hotels while vacationing in Scotland.
> Layovers longer than 90 minutes are unacceptable for domestic flights.

The first directive imposes requirements on accommodations during vacations in a given area. The second prohibits flights with long layovers. Here, we use natural language renderings of advice to aid understandability, but it is easy to map to our structured model. For the first example, the target activity is defined as operations with feature Vacation, and with role Location filled by Scotland. The advice dictates that the filler for the role Accommodation be a 3-star hotel with ensuite facilities.

3.2 Method Advice

Method advice imposes restrictions on the approaches that can be used in solving a goal or class of goals. It is characterized by the template: `<Use/Don't use> <advised-activity> for <target-activity>`. Thus, method advice consists of target and advised activities, along with a polarity expressing prescription or proscription. For example:

> Find a package bike tour starting in Athens for the vacation in Greece.
> Don't fly between cities less than 200 miles apart.

The first piece of method advice declares that the approach used for a particular portion of the trip should have certain features (*i.e.*, Bike, Package) and role constraints (*i.e.*, start location is Athens). The second specifies restrictions on the approach to be taken for solving a class of transport goals.

3.3 Observations

Advice can be either *abstract* or *grounded*. The former constitutes recommendations that apply to a class of goals and operations; the latter provides recommendations relative to a specific goal or activity. For instance, the advice Use TWA for transatlantic flights is abstract, relating to travel for a class of destinations; in contrast, the advice Use TWA to fly to London relates to a specific city (and hence, a specific travel goal).

Ascertaining whether a piece of advice is satisfied requires more than an examination of the final set of partially-ordered actions that are generally viewed as

'the plan'. As an illustration, consider the advice Stay in 3-star ensuite hotels while vacationing in Scotland, in the context of a trip that includes both business and holiday travel. A final plan for this trip would consist of a set of actions at the level of movements to destinations, stays in accommodations, and tours of various sights. Direct examination of a particular accommodation action in the final plan will not reveal its purpose (business or pleasure); hence, it is not possible to verify that the supplied advice has been followed. In general, verification of advice satisfaction requires examination of the overall problem-solving context in which planning decisions were made.

3.4 Parsing Issues

The examples above illustrate that a range of different surface forms can be mapped into the advice templates. As noted earlier, the *advice parser* is responsible for this mapping. While a discussion of advice parsing is beyond the scope of this paper, we briefly mention some relevant issues.

Advice parsing will generally involve domain-specific interpretation of words to extract role-fills and activities, and to identify use prescription or prohibition. For example, interpretation of Stay in 3-star hotels in Scotland as advice requires an understanding that Stay corresponds to a use prescription.

Roles may not be named explicitly in advice. In the advice Use JAL to fly to Japanese destinations, the role Destination is explicit but the role Carrier is not. In many cases, inference can be used to extract implicit roles. For instance, since JAL is an air carrier, its role could be deduced by identifying variables of the appropriate type in air-travel operators.

Surface-level advice such as Travel first-class on trains can be interpreted in multiple ways. The most basic interpretation is as a directive to employ a particular approach for tasks that involve train travel (should the need for such transportation arise). A more complex interpretation would further attribute an implicit goal to use trains (say, rather than driving). The former corresponds to strategic advice, the latter to a combination of strategic and task advice. The intended interpretation is impossible to ascertain without additional information. Because our focus here is on strategic advice, we ignore any such implicit task designations.

4 Planning Model

Our definition of advice satisfaction requires a model for the underlying planning framework. We employ a Hierarchical Task Network (HTN) model, based loosely on that in [4].

4.1 HTN Planning

The cornerstones of HTN planning are *task networks* and *operator schemas*. Informally, a task network is a partially-ordered set of tasks (goals and actions)[1] to be achieved, along with conditions on the world state before and after tasks are executed. Operator schemas specify methods for reducing an individual goal to some new set of subgoals and actions, under appropriate conditions. HTN planning consists of taking a description of an initial world state, an initial task network, and a set of operator schemas for goal refinement, and then repeatedly applying operator schemas until no further reductions are possible. Each such refinement may contribute additional task-ordering and world constraints to the successive HTNs.

Formally, we define a task network $\tau = \langle T, L, W \rangle$, where T is a set of tasks, L is a set of ordering constraints on tasks, and W is a set of world constraints. Tasks can be either *primitive* or *nonprimitive*, with the former having no possible further refinements. We say that a task network is primitive when it contains only primitive tasks. In this document, we use the terms *task network* and *plan* interchangeably; *partial plan* is used to designate a nonprimitive task network.

An HTN planning problem $\mathcal{P} = \langle \mathcal{O}, \tau_0, S_0 \rangle$ is modeled as a set of operator schemas $\mathcal{O}$, an initial task network τ_0, and a set of propositions S_0 denoting the initial world state. An operator schema is characterized by its purpose $Purpose(O)$ (*i.e.*, the goals to which it can be applied), the preconditions for applying the schema $Preconds(O)$, and the task network $Tasks(O)$ to which a goal matching the purpose can be reduced by applying the schema. The world state, goals, purpose, and preconditions are modeled using a first-order language $\mathcal{L} = \langle Vars, Preds, Consts \rangle$. A solution to an HTN problem is a refinement of the original task network that contains only primitive tasks, provided that all constraints in the task network can be *resolved*. We call the resultant resolved task network a *completed* task network.

To enable representations of strategic advice, an HTN domain is extended to include declarations of *feature* and *role* information for operator schemas. Such declarations constitute metalevel information about the domain and generally are not captured in standard planning models. We use the notation $Features(O)$ to designate the features of an operator schema; role information is captured by a *role-resolution* function $RoleVar(O, R)$ that maps an operator schema and a role to the variable that implements the role in the schema (if one exists). Advice is represented in a metalanguage defined over $\mathcal{L}' = \langle Vars, Preds, Consts, Features, Roles \rangle$.

[1] We adopt an action-oriented rather than state-based interpretation of goals, as is consistent with HTN planning.

An AP problem $\mathcal{AP}$ generalizes an HTN problem to include a set of advice $\mathcal{A}$; that is, $\mathcal{AP} = \langle \mathcal{O}, \tau_0, S_0, \mathcal{A} \rangle$. Informally, a plan is a solution to an AP problem iff it is a solution to the underlying HTN problem and further *satisfies* the advice in $\mathcal{A}$ (as formalized in Section 6). Thus, advice acts as a filter on the set of acceptable solutions to a given planning problem.

4.2 Plan Refinement Structures

As noted above, advice imposes restrictions not only on the completed plan but also on the task refinement process by which the plan is derived. For this reason, a model of advice satisfaction must reference the overall context in which plan refinement decisions were made. We define the *(partial) plan refinement structure* for a (partial) plan (similar to the *hierarchical task network* for a plan in [8]) to be $\pi = \langle \mathcal{P}, \mathcal{N}, \mathcal{D} \rangle$ where $\mathcal{P}$ is the set of task networks produced, $\mathcal{N}$ is the set of nodes in any of the task networks, and $\mathcal{D}$ defines a directed acyclic graph of the refinement relations from a node to each of its descendants.

Each node in a plan refinement structure has attributes defined by its associated task network. One key attribute for processing advice is the partial world model $State(n)$, which captures the relations that necessarily hold prior to the execution of the node's task [1]. Additional node attributes include the goal/task for the node $Goal(n)$, the operator schema that has been used to expand that node $OprSchema(n)$, and the operator bindings for the expansion $\sigma(n)$. $Desc(n)$ designates a node and its descendants.

5 Advice Representation Language

In this section, we define a formal language in which to represent strategic advice, and show how instances of the advice templates map into these representations. For simplicity, we assume a common language of features, roles, and relations for the representations of normalized advice and the planning domain, thus obviating the need for a translation step between them.

Representations for advice are constructed from *role restrictions* and *generalized activities*.

Role Restrictions A role restriction $\langle R, F[x] \rangle$ consists of a *role identifier* R from the set of role symbols *Roles*, and a *filler* $F[x]$ represented as a propositional formula defined over variable x. This formula is composed from the language $\mathcal{L}$ of the underlying planning domain.

Generalized Activities A generalized activity is a pair $< \mathcal{F}, \mathcal{R} >$ consisting of a set of required and prohibited operator features $\mathcal{F} = \mathcal{F}^+ \cup \mathcal{F}^-$, and a set of role restrictions $\mathcal{R}$. A generalized activity does not necessarily map to an instance of any particular operator. In particular, a generalized activity may include role restrictions that span multiple levels of task refinement (*i.e.*, operator application). As such, a generalized activity represents an abstract specification of a plan wedge.

Role and Method Advice As described in Section 3, role advice consists of one or more role-fill descriptions, a target activity, and a polarity (prescribing positive or negative use). Similarly, method advice is characterized as an advised activity, a target activity, and a polarity. A piece of role advice with role-fill ρ and target activity α_G is translated into the representation $(ROLE^+ \ \rho \ \alpha_G)$ if the advice is positive, and $(ROLE^- \ \rho \ \alpha_G)$ if negative. Similarly, a piece of method advice with advised activity α_M and target activity α_G is represented as $(METHOD^+ \ \alpha_M \ \alpha_G)$ if the advice is positive, and $(METHOD^- \ \alpha_M \ \alpha_G)$ if negative.

6 Advice Satisfaction

We now proceed with the definition of *satisfaction* for advice, which is grounded in plan refinement structures for primitive task networks. Overall, we say that a plan refinement structure satisfies a piece of advice if each node in the structure satisfies the advice. Satisfaction of advice by a node is defined below, building on definitions for satisfaction for role restrictions and matches for generalized activities.

Here, we adopt certain notational conventions. We write $v : c \in \sigma$ to indicate that variable v is bound to the value c in the substitution σ. Instantiations of an operator schema O or formula ϕ for bindings in σ are written as O^σ and ϕ^σ. The notation $\overline{\phi}$ is used to represent the logical complement of a formula ϕ. Finally, we define $Opr(n) = OprSchema(n)^{\sigma(n)}$.

Satisfaction of a role restriction ρ by a node requires that the variable that models the role for the node's operator be bound to a term that satisfies the fill constraints in ρ.

Definition 1 (Role Restriction: Node) *Let n be a node in a plan refinement structure with O =* OprSchema(n), *and let ρ be a role restriction $\langle R, F[x] \rangle$. Then:*

- n violates ρ *iff* RoleVar(O, R) $= v$ *is defined,* $v : c \in \sigma(n)$, *and* State(n) $\not\models F[c]$,
- n directly satisfies ρ *iff* RoleVar(O, R) $= v$ *is defined,* $v : c \in \sigma(n)$ *and* State(n) $\models F[c]$.

Note that a node n may neither directly satisfy nor violate a role restriction. Such a situation arises when

there is no variable in $O = OprSchema(n)$ that models the designated role (*i.e.*, $RoleVar(O, R)$ is undefined).

Building on the above definition, we define satisfaction of a role restriction by a plan wedge. We distinguish two categories of satisfaction, *strong* or *weak*, corresponding to the cases where some descendant node does or does not directly satisfy the role restriction.

Definition 2 (Role Restriction: Wedge) *Let n be a node in a plan refinement structure with plan wedge W_n. Then W_n* weakly satisfies *the role restriction ρ iff no node in* $\text{Desc}(n)$ *violates ρ. Also, W_n* strongly satisfies *ρ iff there is some node in* $\text{Desc}(n)$ *that directly satisfies ρ, and no node that violates ρ.*

A node is said to *match* a generalized activity if its operator schema satisfies the feature requirements and its plan wedge satisfies the role restrictions for the activity. Intuitively, a generalized activity describes conditions for an actual operation; hence, the strong version of satisfaction for role restrictions is imposed (thus ensuring that the specified roles are defined and properly filled).

Definition 3 (Match: Feature Set) *An operator schema O* matches *a set of feature specifications $\mathcal{F} = \mathcal{F}^+ \cup \mathcal{F}^-$ iff $\mathcal{F}^+ \subseteq \text{Features}(O)$ and $\mathcal{F}^- \cap \text{Features}(O) = \emptyset$. A goal g* matches *$\mathcal{F}$ iff there is some operator schema $O \in \mathcal{O}$ that matches $\mathcal{F}$ for which* $\text{Purpose}(O)$ *matches g.*

Definition 4 (Match: Generalized Activity) *A node n in a plan refinement structure* matches *a generalized activity $\langle \mathcal{F}^+ \cup \mathcal{F}^-, \mathcal{R} \rangle$ iff* $\text{OprSchema}(n)$ *satisfies $\mathcal{F}^+ \cup \mathcal{F}^-$ and W_n strongly satisfies every role restriction in $\mathcal{R}$.*

Satisfaction of positive role advice by a plan structure node requires that either the node not match the target activity, or the advised role restrictions are satisfied by that node or one of its descendants. Here, we require only weak satisfaction for the advised role constraints, which better matches the intuitive character of role advice. For negative role advice, no descendant of a node matching the target activity should directly satisfy the role restriction.

Definition 5 (Role Advice) *A node n in a plan structure* weakly (strongly) satisfies $(\text{ROLE}^+\ \rho\ \alpha_G)$ *iff either n does not match α_G, or W_n weakly (strongly) satisfies ρ. Also, n* satisfies $(\text{ROLE}^-\ \rho\ \alpha_G)$ *iff n does not match α_G, or no $n' \in \text{Desc}(n)$ directly satisfies ρ.*

The weak and strong versions of satisfaction for positive role advice are equivalent when the roles in the advice are necessarily defined for the wedge underneath the node matching the target activity. This condition often holds. For instance, consider the advice Use TWA for transatlantic flights. All plans for such flights will involve one or more operators that contain the roles Destination and Carrier. We refer to this condition of guaranteed existence in the expansion as *role comprehensiveness.*

Definition 6 (Role Comprehensiveness) *A role R is* comprehensive with respect to a goal *g iff every expansion of g contains a node n such that* $\text{RoleVar}(\text{OprSchema}(n), R)$ *is defined.*

Proposition 1 *Let n be a node in a plan refinement structure with plan wedge W_n. If role R is comprehensive with respect to* $\text{Goal}(n)$ *then W_n weakly satisfies a role restriction ρ iff it strongly satisfies ρ.*

Role comprehensiveness guarantees the duality of $ROLE^-$ and $ROLE^+$, as captured by the following proposition. This result is significant in that it enables positive and negative role advice to be processed in a uniform manner when *role comprehensiveness* holds.

Proposition 2 *If role R is comprehensive with respect to goals that match the feature set of α_G then* $(\text{ROLE}^-\ \langle R, F[x] \rangle\ \alpha_G)$ *is satisfied precisely when* $(\text{ROLE}^+\ \langle R, \overline{F}[x] \rangle\ \alpha_G)$ *is satisfied.*

Satisfaction for negative method advice is straightforward: it requires that no descendant of a node matching the target activity matches the advised activity.

Definition 7 (Method Advice: Negative) *A plan structure node n* satisfies $(\text{METHOD}^-\ \alpha_M\ \alpha_G)$ *iff n does not match α_G, or there is no planning node $n' \in \text{Desc}(n)$ that matches α_M.*

Satisfaction for positive method advice presents a more complex case. Consider an approach that requires only that any node matching the target activity have some descendant node that matches the advised activity. Such a semantics is unsuitable for advice such as Fly between locations further than 200 miles apart; given this advice, a traveler would be unhappy with a plan in which he or she did not fly both legs of a trip from Boston through Chicago to Seattle. At the other extreme, the advice should not necessarily apply 'everywhere': consider a trip that includes a destination D that is both inaccessible by air and more than 200 miles from any other location to be visited. In this case, it should be acceptable to generate a plan that involves driving to D.

We adopt the middle ground for satisfaction of positive method advice: given a node that matches the target activity, it requires both the existence of some descendant node that matches the advised activity and a restriction that operators matching the advised activity are applied to the 'maximum extent possible'. We believe that this definition best matches user intent for method advice.

Definition 8 (Method Advice: Positive) *A plan structure node n* satisfies (METHOD$^+$ α_M α_G) *iff n does not match α_G, or the following conditions hold:*

- *there is some node $n' \in$ Desc(n) that matches α_M* (Existence),
- *no descendant of n that does not match α_M could be replaced by a node that does* (Prescription).

It is not the case that a node satisfies ($METHOD^-$ α_M α_G) iff it does not satisfy ($METHOD^+$ α_M α_G). For example, a node can fail to satisfy the positive version in situations where there are multiple descendant nodes for which an operator matching the advised activity could be applied, and some but not all are used. In this case, the negative version of the advice is also unsatisfied.

Advice satisfaction for a plan refinement structure is defined as follows.

Definition 9 (Advice Satisfaction) *A plan structure node satisfies the set of advice $\mathcal{A}$ iff it satisfies each piece of advice in $\mathcal{A}$. A plan structure π satisfies $\mathcal{A}$ iff each node in π satisfies $\mathcal{A}$.*

We note that our definitions of satisfaction for both role and method advice are *nondirectional* in that the conditions of the target activity and the role or method prescriptions mutually constrain each other. For example, consider the advice Don't drive in cities with more than 300,000 people. In a partial itinerary that includes a visit to a city with more than 300,000 people, this advice prohibits the use of a personal automobile for local transport. In addition, it would restrict the choice of cities to visit for time periods in which a rental car has already been secured.

7 Advice Enforcement

Strategic advice acts as a filter on the set of solutions to a planning problem. As such, it is straightforward to define an algorithm that generates advice-satisfying plans: use a Generate-and-Test scheme with a systematic plan-generation engine and a filter that validates the advice satisfaction conditions. Such an approach is impractical for nontrivial domains because of the size of the underlying search space. Instead, an algorithm is required in which advice *informs* the search process. The challenge is to define techniques in which advice influences local planning decisions (the choices for variable instantiations, operator selections, *etc.*), in order to minimize backtracking that results from unsatisfied advice. Such backtracking will necessarily arise: since the definition of advice satisfaction refers to a completed plan refinement structure, planning choices must be made before their correctness with respect to advice satisfaction can be determined.

Here, we define an HTN-style planning algorithm called PSA *(Plan-generation with Strategic Advice)* that enforces advice. The PSA algorithm is proven *sound* in that it produces only plans that satisfy strategic advice, and *complete* in that it will find such a plan if one exists (given certain assumptions).

In essence, PSA works by adding *advice constraints* to nodes in a task network. Advice constraints are determined by considering the operator choices and world state for an HTN node, along with the context of the current partial plan refinement structure. Advice constraints are operationalized to a combination of restrictions on the use of operators, and *planning constraints* formulated exclusively in the language of the basic domain. With this reduction, advice can be enforced using the operator applicability and constraint testing procedures of the underlying planner. This approach makes it possible for advice processing to be layered on top of the core planning system, yielding a modular and portable implementation. Furthermore, the PSA algorithm can be readily adapted to any operator-based goal-refinement framework, including both generative planners and reactive plan execution systems (such as PRS [12]).

The remainder of this section describes PSA. It begins with the presentation of a high-level description of the HTN algorithm that underlies PSA. An overview of PSA is provided next, followed by a detailed description of the algorithm.

7.1 HTN Algorithm

Standard HTN planning can be characterized as a recursive algorithm on task networks. When the network contains only primitive tasks, the algorithm returns a *completion* of the task network if one is defined, else the algorithm fails. When the network contains nonprimitive nodes, one is expanded by selecting an operator schema that matches the unsolved goal on the node and whose preconditions are satisfiable, applying it to generate an expanded task network, then recursively invoking the algorithm on the new network.

Figure 2 outlines a traditional HTN algorithm (based loosely on [4]). It assumes the following standard planning capabilities. The function *Complete*(τ) produces a completion of the task network τ if one exists (*i.e.*, resolves all conflicts and instantiates variables). The function *SelectOpr*($\mathcal{O}, n, \tau$) nondeterministically chooses an instantiation of an operator schema from the set $\mathcal{O}$ that can be applied to the goal for node n in task network τ. This function first identifies the schemas in $\mathcal{O}$ whose purpose matches the goal for n, using the function *Match*(g_1, g_2). Any one of those schemas whose preconditions are satisfied for the relevant bindings, as computed by *Satisfied*(*Preconds*$(O)^\sigma, n, \tau$), can be returned. The function *Reduce*(n, O^σ, τ) refines a task network by ex-

```
Solve(P = ⟨O, S0, τ0⟩, τ = ⟨T, L, W⟩)
  If τ is primitive
    Return Complete(τ) if defined,
    Else FAIL
  Else for a nondeterministically selected nonprimitive node n ∈ T
    /* Nondeterministically select an applicable operator */
    O^σ ← SelectOpr(O, n, τ)
    /* Apply the selected operator */
    τ' ← Solve(P, Reduce(n, O^σ, τ))
    Return τ'

SelectOpr(O, n, τ)
  Φ1 ← {⟨ Oi, σi⟩ | Oi ∈ O, σi = Match(Goal(n), Purpose(Oi)) and σi ≠⊥}
  Φ2 ← {⟨ Oi, σi'⟩ | ⟨ Oi, σi⟩ ∈ Φ1, σi' = Satisfied(Preconds(Oi)^σi, n, τ) and σi' ≠⊥}
  Return Oi^σi' for some nondeterministically chosen ⟨Oi, σi'⟩ ∈ Φ2
```

Figure 2: Traditional HTN Algorithm

panding a node n by the (possibly partially) instantiated plot of the operator O, adding ordering and world constraints as appropriate.

7.2 The PSA Algorithm

The PSA algorithm, presented in Figures 3 and 4, extends the standard HTN algorithm in a few key ways to support advice. These extensions consist of a modified operator selection process, validation of positive method advice for plan wedges, and the propagation of advice-processing information across refinement levels.

7.2.1 Overview

Before discussing the algorithm in detail, we consider the stages that a given piece of advice passes through, namely *activation* and *enforcement*.

The definition of advice satisfaction by a plan refinement structure requires that each piece of advice be satisfied by every node in the structure. For a given node, many pieces of advice are trivially satisfied in the sense that they do not match the context specified by the target activity of the advice. We say that a piece of advice is *triggered* by a planning node when it matches the target activity feature set for the advice.

Triggered advice is generally not directly enforceable; matches for the advised feature set (for method advice) and all roles must be found before any actions can be taken. Such matching may involve operator applications distributed over multiple refinement (and abstraction) levels. We say that a piece of triggered advice has been *fully-activated* when all such matches have been made; otherwise, we say that the advice is *partially-activated*.

Enforcement of fully-activated advice draws on a combination of techniques. For role advice, enforcement reduces to the addition of planning constraints to the applicability conditions of operators. Method advice additionally restricts the set of candidate operators for a goal based on information about operator features. Finally, positive method advice requires a post-planning verification of the *existence* condition.

Because both the matching required for activation and the resultant enforcement constraints can span multiple refinement levels, PSA employs an approach in which each node in an HTN has an associated set of partially-activated advice and advice constraints. This information is propagated downward in the plan refinement structure as appropriate.

7.2.2 Notation

The PSA algorithm uses the following notation. ϕ_a^A denotes a formula representing the advised role constraints for the advice a; if there are no such constraints, ϕ_a^A is simply TRUE. ϕ_a^T is defined similarly but for the target role constraints. In cases where the advice is unambiguous, we simply write ϕ^A and ϕ^T. Role constraint formulas are defined in terms of a particular role, which will map to some variable in a planning operator. The process of identifying this planning variable is referred to as *role resolution*. We use a placeholder variable labeled by the advice and the type of role (target or advised) to represent the planning variable within unresolved role constraint formulas, *e.g.*, $\phi_a^A[x_a^A]$.

The functions $Features_A(a)$ and $Features_T(a)$ return the advised and target feature set, respectively, of the advice a. The functions $MethodAdvice^+(I)$, $MethodAdvice^-(I)$, $RoleAdvice(I)$ take a set of activated (either fully or partially) advice I and return the subsets of positive method, negative method, and role advice, respectively.

$SolveAP(\mathcal{AP} = \langle \mathcal{O}, S_0, \tau_0, \mathcal{A} \rangle, \tau = \langle T, L, W \rangle)$
  If τ is primitive
    Return $Complete(\tau)$ if defined,
    Else FAIL
  Else for a nondeterministically selected nonprimitive node $N = \langle n, I, C \rangle \in T$
    /* Nondeterministically select an applicable operator with its Advice constraints */
    $\langle O^o, I', C' \rangle \leftarrow SelectOpr_A(\mathcal{O}, N, \tau, \mathcal{A})$
    /* Apply the selected operator and pass along advice info */
    $\tau' \leftarrow SolveAP(\mathcal{AP}, Reduce_A(\langle n, I', C' \rangle, O^o, \tau)$
    If O^o and τ' are not well-defined then FAIL
    /* Test Method constraints for this wedge */
    If for each $\alpha \in MethodAdvice^+(I' - I)$
      there is some $n' \in Desc(n)$ for which $W_{n'}$ matches α
        Return τ'
        Else FAIL

Figure 3: Algorithm for Plan-generation with Strategic Advice (PSA)

7.2.3 PSA Details

PSA diverges from standard HTN planning most significantly in the operator selection process. As defined in Figure 4, the function $SelectOpr_A(\mathcal{O}, \langle n, I, C \rangle, \tau, \mathcal{A})$ extends $SelectOpr(\mathcal{O}, n, \tau)$ both to relativize operator selection and application to triggered advice and to propagate relevant activated advice and advice constraints. $SelectOpr_A$ begins likes its nonadvice counterpart, filtering those operators whose purpose does not match the goal of the node under consideration. It also ends similarly, selecting an unfiltered operator whose applicability conditions are satisfied; however, those conditions have been extended to incorporate additional constraints generated by advice processing. The bulk of the function involves this advice processing for the operators that match the goal of the current node: *advice triggering*, *extraction of advice constraints*, and *role resolution*.

Advice processing begins by identifying for each relevant operator O_i any advice that it triggers. Triggering amounts to matching the target activity feature set, as computed by the test $HasFeatures(O_i, Features_T(a))$.

The extraction of advice constraints differs for the cases of role advice, positive method advice, and negative method advice. For role advice, the extracted advice constraint is simply the disjunction $\overline{\phi^T} \vee \phi^A$. Method advice presents a more complex case because it posits higher-order constraints defined over operators and domain objects. As a result, it is not possible to map them to planning constraints in $\mathcal{L}$ that can be directly processed by the underlying planner (as is the case with role advice). Instead, constraints defined over a mixture of operators and domain objects must be managed explicitly at a higher level. We adopt a *case analysis* approach in which satisfaction is considered in turn for the different kinds of constraints (on operators and on domain objects).

Consider first negative method advice. Any operator that does not match the feature set of the advised activity requires no additional constraints. For an operator that does match, either the target role-fill constraints ϕ^T or the advised role-fill constraints ϕ^A must fail. Thus, $\overline{\phi^T} \vee \overline{\phi^A}$ is added to the applicability conditions of the operator.

Positive method advice requires a different approach: either the target activity must not match, or the existence and prescription clauses must both be satisfied. For an operator that matches the feature set of the advised activity, we require that the advised role constraints ϕ^A be satisfied. For an operator that does not match the advised feature set, if there is *some* unfiltered operator that does match, then the target activity must not match. The constraint $\overline{\phi^T}$ guarantees this condition.[2]

Advice constraints are defined over roles; before they can be enforced, the roles must be resolved. The function $ResolveRoles(O, \Delta)$ performs role resolution; it replaces any role variable in the set of constraints Δ with the variable that models that role in the operator schema (if one exists). Resolved constraints (computed by $Resolved(C)$) are added to the applicability constraints of the associated operator. Unresolved constraints (computed by $Unresolved(C)$) are passed to all descendant nodes.

The actual propagation of advice information is straightforward; $Reduce(n, O, \tau)$ is generalized to the

[2]This approach to enforcing positive method advice is overly conservative in that the added constraints may be unnecessary (because some other node in the plan satisfies the *existence* clause) yet they block application of the operator. We return to this point in the discussion of completeness below.

$SelectOpr_A(\mathcal{O}, \langle n, I, C\rangle, \tau, \mathcal{A})$
$\Phi_1 \leftarrow \{\langle O_i, \sigma_i\rangle \mid O_i \in \mathcal{O}, \sigma_i = Match(Goal(n), Purpose(O_i))$ and $\sigma_i \neq \perp\}$

/* Accumulate triggered advice for each candidate operator */
For $\langle O_i, \sigma_i\rangle \in \Phi_1$
$I_i \leftarrow \{a \mid a \in \mathcal{A}$ and $HasFeatures(O_i, Features_T(a))\}$
$I' \leftarrow \Sigma I_i$

/* Extract advice constraints */
For $\langle O_i, \sigma_i\rangle \in \Phi_1$ **/* Initialize advice constraints */**
$R_i \leftarrow \{\}$, $M_i^+ \leftarrow \{\}$, $M_i^- \leftarrow \{\}$

For $a \in RoleAdvice(I_i)$ **/* Extract constraints from role advice */**
$R_i \leftarrow R_i \cup \{\overline{\phi_a^T[x_a^T]} \vee \phi_a^A[x_a^A]\}$

For $m \in MethodAdvice^-(I \cup I')$ **/* Extract constraints from negative method advice */**
$Sat \leftarrow \{\langle O_i, \sigma_i\rangle \mid \langle O_i, \sigma_i\rangle \in \Phi_1$ and $HasFeatures(O_i, Features_A(m))\}$
For $\langle O_i, \sigma_i\rangle \in Sat$, $M_i^- \leftarrow M_i^- \cup \{\overline{\phi_m^A[x_m^A]} \vee \overline{\phi_m^T[x_m^T]}\}$

For $m \in MethodAdvice^+(I \cup I')$ **/* Extract constraints from positive method advice */**
$Sat \leftarrow \{\langle O_i, \sigma_i\rangle \in \Phi_1 \mid \langle O_i, \sigma_i\rangle \in \Phi_1$ and $HasFeatures(O, Features_A(m)\}$
If $Sat \neq \{\}$:
For $\langle O_i, \sigma_i\rangle \in Sat$, $M_i^+ \leftarrow M_i^+ \cup \{\phi_m^A[x_m^A]\}$
For $\langle O_i, \sigma_i\rangle \in \Phi_1 - Sat$, $M_i^+ \leftarrow M_i^+ \cup \{\overline{\phi_m^T[x_m^T]}\}$

/* Collect resolved advice constraints for each Operator */
For $\langle O_i, \sigma_i\rangle \in \Phi_1$
$ResolveRoles(O_i, C \cup R_i \cup M_i^+ \cup M_i^-)$
$C_i \leftarrow Unresolved(C \cup R_i \cup M_i^+ \cup M_i^-)$
$Add_i \leftarrow Resolved(C \cup R_i \cup M_i^+ \cup M_i^-)$

/* Return an operator that satisfies advice constraints along with updated advice info */
$\Phi_2 \leftarrow \{\langle O_i, \sigma_i'\rangle \mid \langle O_i, \sigma_i\rangle \in \Phi_1, \sigma_i' = Satisfied(Preconds(O_i)^{\sigma_i} \cup Add_i^{\sigma_i}, n, \tau)$ and $\sigma_i' \neq \perp\}$
Return $\langle O_i^{\sigma_i'}, I \cup I_i, C_i\rangle$ for some nondeterministically chosen $\langle O_i, \sigma_i'\rangle \in \Phi_2$

Figure 4: Operator Selection for PSA

function $Reduce_A(\langle n, I, C\rangle, O, \tau)$ to pass along advice constraints and activated advice.

As noted above, the existence clause for satisfaction of positive method advice cannot be verified until the wedge beneath the node has been completed. An explicit check of this condition is included as the last step in the PSA algorithm. Since the validation of the condition is straightforward, the details are omitted.

7.3 Discussion

Our presentation of PSA has been biased toward highlighting the reuse of standard HTN operations. For reasons of efficiency, the PSA algorithm should not be implemented directly as presented here. For instance, extraction of role constraints and resolution of role variables should be done only for the selected operator, rather than for all operators matching the current goal prior to the final selection decision. In contrast, extraction of method constraints should be done prior to the selection process because the method constraints can eliminate certain operators up front.

There is a similarity between the kind of *constraint-augmented planning* embodied in the PSA algorithm and Constraint Logic Programming (CLP) [7]: both consist of a problem-reduction search augmented with constraints on the overall structure being defined. For CLP, the constraints restrict instantiations for variables; for advisable planning, the constraints further restrict the choice of problem-reduction rules (*i.e.*, the operator schemas).

7.4 Properties of PSA

It is straightforward to show that the PSA algorithm reduces to standard HTN planning when there is no

advice. The following correctness result also holds.

Proposition 3 (Correctness of PSA) *Application of the PSA algorithm to an advised planning problem* $\mathcal{AP} = \langle \mathcal{O}, S_0, \tau_0, \mathcal{A} \rangle$ *will produce a plan refinement structure that both satisfies* $\mathcal{A}$ *and is a solution to the planning problem* $\langle \mathcal{O}, S_0, \tau_0 \rangle$.

We note that given *role comprehensiveness*, the strong version of satisfaction for role advice follows; otherwise, only weak satisfaction holds. The difference results because PSA folds in constraints extracted from role advice only when the roles for the constraints have been resolved; thus, if the roles are not found, the associated role advice is effectively ignored.

The PSA algorithm may fail to find solutions in certain cases where solutions exist. This incompleteness stems from our conservative approach to enforcing the prescription clause for positive method advice. In particular, the constraints added to enforce this clause are too strong in situations when a node matching the target activity has multiple descendant nodes for which constraints may be added to guarantee matching to the advised activity. As an illustration, consider a situation where there are two descendant nodes to which operators could be applied that match the advised feature set. PSA would force for both nodes the selection of operators that satisfy the advised feature set, even in cases where the use of the operators is mutually inconsistent (for instance, they each may require a consumable resource of which there is only one remaining). However, the definition of satisfaction for positive method advice requires that both be selected *only if it is consistent to do so.*

We can show that the PSA algorithm is complete when there is at most one node for which constraints are added to enforce the prescription clause; we refer to this condition as the *Uniqueness of the Advised Activity (UAA)* for a given piece of positive method advice relative to a set of operators. The UAA condition holds frequently in practice. For instance, in planning a holiday there is generally a single high-level vacation type to select (*e.g.*, bike tour *vs* camping tour *vs* driving tour). Thus, advice such as Use a bike tour for the vacation in California would satisfy UAA, since the choice of tour type would only be made once. Furthermore, the UAA condition is easily verified by straightforward syntactic analysis of operator schemas. However, one can formulate natural problems for which UAA is violated. For instance, one could formulate a multi-phase holiday operator that encompasses several sub-vacations, each of which would require a choice of vacation type.

Proposition 4 (Completeness of PSA) *Application of the PSA algorithm to an advised planning problem* $\mathcal{AP} = \langle \mathcal{O}, S_0, \tau_0, \mathcal{A} \rangle$ *for which UAA holds will produce a plan refinement structure that both satisfies* $\mathcal{A}$ *and is a solution to the planning problem* $\langle \mathcal{O}, S_0, \tau_0 \rangle$, *provided such a plan exists.*

8 Related Work

Until recently, there had been few attempts to develop domain-independent advice-taking systems. Concerns for the usability of AI systems and problems with knowledge acquisition have prompted a marked increase in activity in this area during the past few years.

The TRAINS [5] project seeks to provide users with the means to interactively guide the construction and execution of a plan through a cooperative, mixed-initiative effort. While its overall objectives are similar to ours, the research directions of the two efforts are highly complementary. HCI issues are a major focus for the TRAINS project (*e.g.*, language processing, dialog management, multi-media), while our work emphasizes the identification of advice idioms and corresponding enforcement algorithms.

The *reactive scheduling* model of the DITOPS system [13] also has much in common with our work. At a high level, its *spread-sheet* metaphor provides a good characterization of advisability: the user should be able to specify characteristics of the desired solution and have the system sort out the details. One key difference is that DITOPS focuses on scheduling rather than planning. In addition, it emphasizes changes to domain constraints, while the our work focuses on advice that describes high-level properties of solutions.

The TRAINS and DITOPS projects are similar to our work in that they emphasize *product-related* advice that describes characteristics of the desired end-product. In contrast, *performance-related* advice encodes base-level problem-solving expertise. As such, product-related advice serves as an adjunct to the underlying problem-solving knowledge, while performance advice amounts to an extension of it.

The original work on performance advice is Mostow's [11], which models advice as a high-level description of a task for which there is no explicit method to achieve it. The work focuses on *operationalization* – the transformation of advice (using sound and heuristic methods) into directives that can be executed directly by the problem-solving system being advised. Its notion of advice-taking amounts to 'filling in' gaps in planning operators, in contrast to our approach of restricting how operators should be instantiated and applied.

There has been a recent surge of interest in improving the performance of reinforcement learners through user guidance (see [9] for a good overview). Broadly speaking, that work focuses on the provision of additional domain knowledge to improve the overall performance of a system. Many of these efforts are actually a form of 'programming by example'. However, certain

of them have more of a flavour of performance-related advice-taking, using general-purpose languages to encode the additional domain knowledge [9, 6]. One interesting feature of these efforts is that advice is refined during problem-solving, in response to the success with which it has been applied previously.

9 Conclusions

The notion of problem-solving systems that can take advice from humans has been around since the start of AI [10]. Despite the conceptual appeal, there has been little success to date in building automated advice-taking systems because of the intractability of the task in its most general form. We believe that the paradigm can be made tractable for specific classes of applications and tasks by grounding advice in a focused set of problem-solving activities. The research described here presents a step toward this goal for the paradigm of generative planning. Its primary contributions are the presentation of an advice-taking framework for AI planning systems, the formalization of strategic advice, and the definition of a sound and complete HTN planning algorithm for enforcing strategic advice.

We have built an initial Advisable Planner prototype that implements our theory of strategic advice. The system consists of an advice manager layered on top of SIPE–2, a mature HTN planner [15, 16]. Intrusions into the underlying code were minor: for the most part, the advice processing is completely separate from the core planning capabilities. Our prototype has been used to enforce both travel planning advice similar to the examples presented in this paper and advice for a crisis-action planning domain [17]. Currently, we are applying the system to air-campaign planning, with the goal of encouraging nonexperts to embrace AI planning technology.

Immediate next steps for this work are to develop richer idioms for strategic advice, and to define comparable formal models and enforcement algorithms for other categories of advice (including evaluational and task advice). Further down the road, we intend to explore utility models for partial satisfaction of advice that will allow intelligent trade-offs to be made among sets of conflicting advice.

Acknowledgements

This research has been supported by DARPA Contract F30602-95-C-0259. The author would like to thank David Wilkins for providing insight into the workings of SIPE–2.

References

[1] D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333–378, 1987.

[2] K. Currie and A. Tate. O-Plan: the open planning architecture. *Artificial Intelligence*, 32(1), 1991.

[3] B. Drabble and A. Tate. O-Plan mixed initiative planning capabilities and protocols. Technical Report 24, University of Edinburgh, 1995.

[4] K. Erol, J. Hendler, and D. S. Nau. Semantics for hierarchical task-network planning. Technical Report CS-TR-3239, Computer Science Department, University of Maryland, 1994.

[5] G. Ferguson, J. Allen, and B. Miller. TRAINS-95: Towards a mixed-initiative planning assistant. In *Proceedings of the Third International Conference on AI Planning Systems*. AAAI Press, 1996.

[6] D. Gordon and D. Subramanian. A multistrategy learning scheme for agent knowledge acquisition. *Informatica*, 17:331–346, 1994.

[7] J. Jaffar and J.-L. Lassez. Constraint logic programming. In *ACM Symposium on Principles of Programming Languages*, 1987.

[8] S. Kambhampati and J. Hendler. A validation-structure-based theory of plan modification and reuse. *Artificial Intelligence*, 55(2):192–258, 1992.

[9] R. Maclin and J. W. Shavlik. Creating advice-taking reinforcement learners. *Machine Learning*, 22:251–282, 1996.

[10] J. McCarthy. Programs with common sense. In *Symposium on the Mechanization of Thought Processes*, pages 77–84, 1958.

[11] D. J. Mostow. *Mechanical Transformation of Task Heuristics into Operational Procedures*. PhD thesis, Computer Science Dept., Carnegie-Mellon University, 1981.

[12] K. L. Myers. *User's Guide for the Procedural Reasoning System*. Artificial Intelligence Center, SRI International, Menlo Park, CA, 1993.

[13] S. F. Smith and O. Lassila. Toward the development of flexible mixed-initiative scheduling tools. In M. H. Burstein, editor, *ARPA/Rome Laboratory Planning and Scheduling Initiative Workshop Proceedings*. Morgan Kaufmann, February 1994.

[14] M. Stefik. Planning and meta-planning. *Artificial Intelligence*, 16(2), 1981.

[15] D. E. Wilkins. *Practical Planning: Extending the Classical AI Planning Paradigm*. Morgan Kaufmann, 1988.

[16] D. E. Wilkins. *Using the SIPE-2 Planning System: A Manual for Version 4.3*. Artificial Intelligence Center, Menlo Park, CA, August 1993.

[17] D. E. Wilkins and R. V. Desimone. Applying an AI planner to military operations planning. In M. Fox and M. Zweben, editors, *Intelligent Scheduling*. Morgan Kaufmann, 1994.

Efficiency Competition through Representation Changes: Pigeonhole Principle vs. Integer Programming Methods

Yury V. Smirnov
Computer Science Dept.
Carnegie Mellon University
Pittsburgh, PA 15213-3891, USA
smir@cs.cmu.edu

Manuela M. Veloso
Computer Science Dept.
Carnegie Mellon University
Pittsburgh, PA 15213-3891, USA
mmv@cs.cmu.edu

Abstract

The Pigeonhole Principle (PHP) has been one of the most appealing methods of solving combinatorial optimization problems. Variations of the Pigeonhole Principle, sometimes called the "Hidden" Pigeonhole Principle (HPHP), are even more powerful and often produce the most elegant solutions to nontrivial problems. However, some Operations Research approaches, such as the Linear Programming Relaxation (LPR), are strong competitors to PHP and HPHP. They can also be applied to combinatorial optimization problems to derive upper bounds. It has been an open question whether PHP or LPR establish tighter upper bounds and how efficiently, when applied to the same problem. Challenged by this open question, we identify that the main reason for the lack of ability to compare the efficiency of PHP and LPR is the fact that different problem representations are required by the two methods. We introduce a problem representation change into an Integer Programming form which allows for an alternative way of solving combinatorial problems. We also introduce several combinatorial optimization problems, and show how to perform representation changes to convert the original problems into the Integer Programming form. Using the new problem model, we re-define the Pigeonhole Principle as a method of solving Integer Programming problems, determine the difference between PHP and HPHP, prove that PHP has the same bounding power as LPR, and demonstrate that HPHP and Integer cuts are actually similar representation changes of the problem domains.

1 INTRODUCTION

The Pigeonhole Principle (PHP) [Thucker, 1980] is traditionally considered the simplest and most elegant method of deriving tight upper bounds for a class of combinatorial optimization problems. In its original formulation, the Pigeonhole Principle (also known as the Dirichlet drawer principle) states that "it is impossible to place $N + 1$ pigeons in N holes so that there is at most one pigeon in each hole." In this simple form, PHP looks like a naïve kindergarten-level rule. However, if taken to a higher level of different number of objects (pigeons) allowed in abstract units (holes), or used as a part of a multi-step logical proof, it quickly loses the nuance of obviousness. Though PHP has no clear-cut mathematical origin, its applications involve some of the most profound and difficult results in combinatorics of great relevance to Artificial Intelligence.

To solve a combinatorial optimization problem by PHP, one needs to represent a problem in such a way that the principle may be applied, i.e. to perform what we call the representation change. For example, to identify what in the problem should be mapped to objects (pigeons) and units (holes). For simple problems, the applications of PHP are almost straightforward and are obtained directly from the nature of the problems. However, often the proof by PHP requires additional heuristic knowledge that allows the effective representation change. If this representation change is found, then PHP easily produces the tight upper bound and an optimal solution. For some combinatorial optimization problems it is a very challenging task to find such a representation change. Hardness of this task encourages to look for alternative methods.

The introduction of polynomial-time algorithms solving Linear Programming problems allows some Operations Research methods to be applied also to combinatorial optimization problems to establish upper bounds for such problems. For example, the Linear Programming Relaxation (LPR) method in conjunction with Integer cuts has been applied effectively to solve some Integer Programming (IP) problems. LPR can be seen as requiring less effort to apply than PHP, because in general, unlike PHP, it does not need any additional knowledge or representation changes to provide upper bounds for IP problems.

The two approaches, namely PHP and LPR-based methods can be seen as "competitors" for solving combinatorial

problems. It has been an open question[1] whether which method, if any, can provide better upper bounds and which method, if any, is more efficient, when applied to the same combinatorial optimization problems. This paper reports on our work in solving this open question. As hinted so far, our work analyzes carefully the problem representation issues involved in the two approaches. We formally introduce an appropriate representation change that makes the comparison and analysis possible.

There has been attempts in comparing the two mentioned approaches, see [Ginsberg, 1996]. We chose an Empirical way of re-defining the Pigeonhole Principle, because it allowed us to apply it to a series of problems of a great importance for Artificial Intelligence. We also found an alternative meaning of representation changes that in conjunction with the original PHP constitute the "Hidden" Pigeonhole Principle.

For some problems, like the "Mutilated" Checkerboard problem [Newell, 1965], immediate applications of the Pigeonhole Principle fail to provide the tight upper bound. Additional knowledge (heuristics) or representation changes are needed to accomplish a successful application of PHP that would establish the best feasible value for a given problem. Had this knowledge been provided in advance or, equivalently, had the problem been stated in a more suitable (for PHP) form, such an application would be easy and almost straightforward. However, the process of obtaining relevant knowledge or finding appropriate representation changes is problem-dependent, and often is a challenging task itself. Successful applications that combine acquiring relevant knowledge or performing representation changes and applications of the Pigeonhole Principle are sometimes called the "Hidden" Pigeonhole Principle (HPHP), though the difference between PHP and HPHP has never been clearly defined. Problems solved by HPHP are usually very non-trivial, and HPHP is often the most elegant and simple way of solving such problems. In Section 4 we discuss the "Mutilated" Checkerboard and Firm Tiling problems in detail and demonstrate the similarity between the Hidden Pigeonhole Principle and the combination of LPR with Integer cuts.

Our work includes:

1. Re-defining PHP as a method of solving Integer Programming problems.
2. Proving that both PHP and LPR obtain the same upper bounds.
3. Providing an alternative approach to solving combinatorial optimization problems for which the effective representation change required by PHP is hard to find. The alternative approach that can be implemented in automated logical proof, AI planning, or scheduling systems, consists of the following steps:
   (a) Convert a combinatorial optimization problem into an Integer Programming form.
   (b) Apply LPR to obtain an upper bound B which, as we proved, is the same as the upper bound obtained by PHP.
   (c) Construct an optimal solution of value B.

In this work, we also draw a clear splitting line between the original and the "Hidden" Pigeonhole Principles, analyze the latter one and compare its bounding power with that of Integer cuts.

2 OUR REPRESENTATION CHANGE APPROACH

The Pigeonhole Principle is the most simpleminded ideas imaginable, yet its generalizations involve some of the most profound results in combinatorics[Thucker, 1980]. A more sophisticated fashion of applying PHP is sometimes called the "Hidden" Pigeonhole Principle. In such cases, an optimal solution is usually nontrivial and the application of the Hidden Pigeonhole Principle (HPHP) is probably the simplest way of proving the optimality of a known solution.

Linear Programming Relaxation (LPR) is one of the most popular Operations Research methods of establishing upper bounds for Integer Programming problems[Nemhauser and Wolsey, 1985]. Its main calculation engine is based on the discovery of polynomial methods of solving Linear Programming problems, such as the Ellipsoid algorithm[Khachian, 1979]. Actually, LPR is a two-step procedure consisting of relaxing integer requirements and solving the corresponding Linear Programming problem.

Simplisticly, LPR can be viewed as a black-box with a particular instance of an Integer Programming problem as an input and a calculated upper bound as an output. From this point of view, LPR looks preferable to PHP, because it does not need any heuristic knowledge in deriving upper bounds. LPR can be applied to any instance of an Integer Programming problem without additional representation changes. Thus, the main questions of the competitive analysis between LPR and PHP can be stated as the *knowledge representation problem*: Which form of presenting combinatorial optimization problems allows to establish tighter bounds?

We start the following sections with the discussion of Linear Programming techniques relevant to our subject, then we re-define the Pigeonhole Principle for arbitrary instances of Integer Programming problems, determine the difference between PHP and HPHP, compare computational power of PHP and LPR, overview similarities between HPHP, PHP with heuristics, and Integer cuts for Integer Programming problems. We illustrate the discussion by a series of combinatorial optimization problems of a gradually increasing complexity.

[1] Identified at the First International Workshop on AI and OR, Portland, OR, June, 1995.

2.1 Important Facts from Linear Programming

In this section we introduce basic definitions and facts from the theory of Linear and Integer Programming. Readers familiar with this subject may skip this section.

A particular instance of a Linear Programming (LP) problem consists of the goal function and the set of inequalities (constraints). Both the goal function and the constraints depend linearly on each variable. Throughout this paper we consider variables to be real-valued, and their number is finite, the number of inequalities is also finite. Thus, a typical LP problem is of the following type:

$$\text{goal function}: \quad max(\sum_{i=1}^{N} c_i x_i)$$

$$\text{constraint set } J: \quad \sum_{i=1}^{N} a_{ij} x_i \leq b_j \quad j = 1, \ldots, M$$

$$x_i \in R \quad i = 1, \ldots, N$$

Usually coefficients a_{ij}, b_j and c_i are rational, it allows to limit our consideration to rational-valued variables $x_i \in Q \quad i = 1, \ldots, N$. To simplify further discussion, we introduce a particular LP problem which will help to illustrate the discussion:

$$\text{goal function}: \quad max(y) \tag{1}$$

$$\text{constraint set } J: \quad -x \leq 0 \tag{2}$$

$$-y \leq 0 \tag{3}$$

$$y - x \leq 1 \tag{4}$$

$$x + y \leq 3 \tag{5}$$

$$x + 2y \leq 8 \tag{6}$$

$$x \in Q \tag{7}$$

$$y \in Q \tag{8}$$

Figure 1 shows the feasible region, optimal solution $x^* = (1, 2)$, and the goal vector corresponding to the goal function (1). Theory of Linear Programming states that for any optimal solution x^* there exists a subset of constraints, called active constraints, such that:

- Inequalities representing active constraints are actually equalities for x^*, or, equivalently, there is no slack for active constraints with respect to x^*.
- The number of active constraints can vary from 1 to M.
- Goal function is a positive combination of the left-hand sides of active constraints.
- It is always possible to represent the goal function as a positive combination of at most N active constraints.

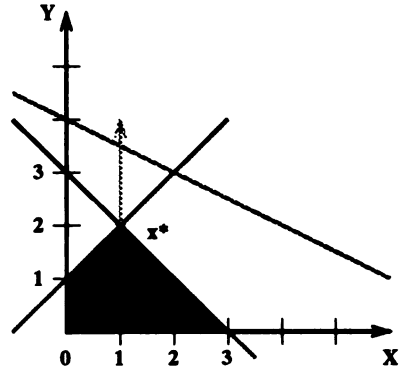


Figure 1: A Particular Instance of a LP Problem

In our example (1-8) inequalities (4) and (5) form the set of active constraints with respect to the optimal solution $x^* = (1, 2)$. Thus, the goal function (1) can be presented as a positive combination of (4) and (5): $y = \frac{1}{2}(y - x) + \frac{1}{2}(x + y)$.

If we vary inequality (6), it may also become an active constraint. For example, inequality $x + 2y \leq 5$ is an active constraint with respect to the optimal solution $x^* = (1, 2)$. If added to the existing set of active constraints (4) and (5), it would constitute a redundancy: Left-hand sides of any two out of three inequalities can be used in a positive weighted sum to obtain the goal function. On the other hand, the goal function can also vary within a certain range to preserve the same optimal solution and the set of active constraints. In problem (1-8) for any goal function of the form $y + \alpha x$ with $\alpha \in [-1, 1]$, the set of active constraints consists of (4) and (5). For the extreme values of $\alpha \in \{-1, 1\}$ one of the active constraints becomes redundant, since the goal function coincides with the left-hand side of the other active constraint. Moreover, in both extreme cases there exists an infinite number of feasible solutions attaining the same optimal value. These optimal solutions form a face of a polyhedron of feasible solutions defined by the IP problem's constraints. For example, in problem (1-8) the set of feasible solutions forms a 2-D tetragon (see Figure 1), and the set of optimal solutions is either a zero-dimensional face (a single 2-D point) or a one-dimensional face (one of the tetragon's sides).

Thus, on one hand, a positive weighted sum of left-hand sides of active constraints establishes an upper bound: Since $\sum_j \sum_i \alpha_j a_{ij} x_i = \sum_i c_i x_i$, where $\alpha_j \geq 0$, and $\sum_i a_{ij} x_i \leq b_j$, then $\sum_i c_i x_i \leq \sum_j \alpha_j b_j$. On the other hand, any feasible solution $\bar{x}$ sets a lower bound for the goal function $\sum_i c_i \bar{x}_i$. Furthermore, active constraints have no slack with repeat to optimal solution x^*, therefore, the lower bound provided by x^* coincides with the upper bound established by the positive weighted sum of active constraints and both are equal to the optimal value $\sum_i c_i x_i^*$.

Integer Programming (IP) problems have an additional requirement that some of its variable are integer-valued. Throughout this paper we consider only those IP problems

that have linear goal functions and linear constraints. In general, the addition of integer-valued variables makes an IP problem NP-hard. However, in some particular cases, it is possible to solve IP problems efficiently, by applying Linear Programming Relaxation (LPR) or a combination of Integer Cuts and LPR[Nemhauser and Wolsey, 1985]. LPR is a simple procedure that consists of two steps:

1. Drop (relax) integrality requirements for all variables.
2. Solve the correspondent LP problem.

For some IP problems, like problem (1-8) introduced earlier, LPR outputs an integer feasible solution, thus, solving such IP problem. In other cases, when LPR outputs a fractional optimal solution x^*, it establishes an upper bound $\sum c_i x_i^*$ for the goal function. In such cases, LPR is usually not very helpful in finding integer solutions or indicating the tightness of the upper bound.

Integer cuts produce additional constraints that utilize the integral nature of a subset of variables. Additional constraints can be derived in many different ways, Integer cuts produce ones that cannot be obtained by taking positive weighted sums of the existing inequalities. In conjunction with LPR, Integer cuts are capable of solving IP problems, whereas the efficiency of this combination in obtaining tight upper bounds relies on the "quality" of newly created constraints constructed by Integer cuts techniques. Examples of the Hidden Pigeonhole Principle's applications, namely the Mutilated Checkerboard problem and the Firm Tiling problem discussed in Section 4, illustrate an alternative way of solving these problems through Chvatal-Gomory's (Integer) cuts and LPR.

2.2 Converting Problems into the Integer Programming Form

We identified that the main reason of the lack of ability to compare the efficiency of PHP and LPR is that different problem representations are required by the two methods. Our work consisted of developing appropriate changes of representations that made possible a competitive analysis of the efficiency of these two methods.

To make both PHP and LPR applicable to the same combinatorial optimization problems, we perform representation changes for each problem discussed in this paper to convert them into Integer Programming problems of the following form:

$$\text{goal function}: \quad max(\sum_{i=1}^{N} c_i x_i)$$

$$\text{constraint set } J: \quad \sum_{i=1}^{N} a_{ij} x_i \le b_j \quad j = 1, \ldots, M$$

$$\text{integrality requirements}: \quad x_i \in Z \quad i = 1, \ldots, N$$

A combinatorial optimization problem is defined as finding a solution $x^* = (x_1^*, \ldots, x_N^*)$ that is feasible, i.e. satisfies all the constraints from J and the integrality requirements, and also attains the best value of the goal function among all feasible solutions.

We considered a series of known combinatorial optimization problems, to which the Pigeonhole Principle is applicable. The empirical evidence obtained through this study suggested that the following definition reflects correctly the combinatorial nature of the Pigeonhole Principle. We confirm the correctness of this definition through a set of problems that illustrate applications of PHP for the original combinatorial form and the Integer Programming reformulation.

Definition 1 *Suppose that an Integer Programming problem of the above type has a known feasible solution x^*. We say that the problem admits a proof by the Pigeonhole Principle if there exists a subset of constraints $\hat{J} \subseteq J$, such that the sum of the left-hand sides of the inequalities from $\hat{J}$ (repetitions are allowed in $\hat{J}$) is a multiple of the goal function*

$$\sum_{j \in \hat{J}} \sum_{i=1}^{N} a_{ij} x_i = k \sum_{i=1}^{N} c_i x_i$$

and the value of x^* equals the same multiple of the sum of the right-hand side constants

$$\sum_{j \in \hat{J}} b_j = k \sum_{i=1}^{N} c_i x_i^* \ .$$

If such a solution x^* is not known, then the best value

$$\min_{\hat{J} \subseteq J} \frac{1}{k} \sum_{j \in \hat{J}} b_j \ , \quad \text{such that} \quad \sum_{j \in \hat{J}} \sum_{i=1}^{N} a_{ij} x_i = k \sum_{i=1}^{N} c_i x_i$$

holds for some positive $k > 0$, is called an upper bound obtained by PHP.

In general, the procedure of considering a subset of constraints and summing them up (possibly with positive weights) results in setting upper bounds for the value of an optimal solution. Any feasible solution provides a lower bound for the optimal value of the problem. We identify applications of PHP with finding a subset of constraints and obtaining the optimal value of the goal function. In this case, the established upper bound matches the lower bound provided by the found optimal solution.

Definition 2 *If a set of constraints added to an Integer Programming problem changes the original IP problem into an extended IP problem that admits a proof by the Pigeonhole Principle, we say that the original problem admits a proof by the Hidden Pigeonhole Principle (HPHP).*

Additional constraints can be obtained in many different ways. Weighted positive sums, for example, can simplify some of the existing constraints, but they would not contribute any restrictions on fractional optimal solutions obtained by Linear Programming Relaxation. In Section 4 we consider an Integer cuts technique that allows to derive tighter valid inequalities and to remain all integer solutions feasible. Constraints obtained from Branch-and-Bound methods can also be sought as an addition to the existing set of constraints. We consider the way of constructing additional constraints and the sanity check that an optimal integer solution has not been cut off to be the responsibility of the problem solver.

The introduced representation change allows to reformulate the original PHP statement with 11 pigeons and 10 holes as the following IP problem:

$$max(\sum_{i=1}^{11}\sum_{j=1}^{10} x_{ij})$$

$$\sum_{i=1}^{11} x_{ij} \leq 1 \quad j = 1, \ldots, 10$$

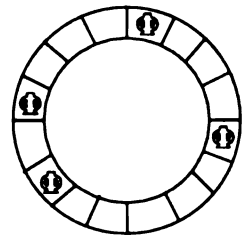
$$x_{ij} \in \{0, 1\},$$

where x_{ij} represents the amount of the ith pigeon in the jth hole. Thus, if we drop integer requirements and sum up all the constraints, we obtain an upper bound: $\sum_{i=1}^{11}\sum_{j=1}^{10} x_{ij} = \sum_{j=1}^{10}\sum_{i=1}^{11} x_{ij} \leq 10$. Since an obvious solution $\{x_{ii} = 1 \text{ for } i = 1, \ldots, 10;\ x_{ij} = 0 \text{ for } i \neq j\}$ provides the same value, we proved it to be optimal. For this simple problem, PHP and LRP approaches are identical.

In most obvious cases, the desired subset of constraints is the whole set of original inequalities. Consider, for example, the problem of placing chess kings on a circular chessboard with even number of squares (see Figure 2). One is supposed to find the maximal number of kings that can be placed on such a board so that no king is attacking another. Recall that a chess king attacks all its adjacent squares. For the board presented in Figure 2, we get the following IP problem:

If we sum up all the inequalities, we get $2\sum_{i=1}^{N} x_i \leq N$, which is equivalent to $\sum_{i=1}^{N} x_i \leq N/2$. The same upper bound can be obtained from applying a combination of two-coloring and PHP: if we color the circular board with even number of squares in alternating black and white colors, we can place kings on either color attaining the optimum value of $N/2$ derived this way by PHP.

The case when a proper subset of constraints is involved in obtaining a tight upper bound is more complicated. To apply the Pigeonhole Principle in the original combinatorial form, one has to find which problem's objects should be matched with pigeons and which - with holes. In the Integer Programming form it means that one has to come up with a rule (heuristic) of finding a desired subset of constraints to sum them up. For some problems, it is easy to find such a subset,



$$max(\sum_{i=1}^{N} x_i)$$

$$x_1 + x_2 \leq 1$$

$$x_2 + x_3 \leq 1$$

$$\ldots$$

$$x_{N-1} + x_N \leq 1$$

$$x_N + x_1 \leq 1$$

$$x_i \in \{0, 1\}$$

Figure 2: Chess kings on a circular chessboard with N positions and corresponding IP representation. Kings cannot be in adjacent positions, i.e. attack each other.

whereas for others it is not obvious. For example, the popular N-Queen problem[Hoffman et al., 1969, Nauck, 1850], which is the problem of placing the maximal number of chess queens on NxN-chessboard, so that no queen is attacking another, can be represented as the following IP problem:

$$max(\sum_{i=1}^{N}\sum_{j=1}^{N} x_{ij})$$

$$\text{for each } i \in \{1, \ldots, N\} \text{ and } j \in \{1, \ldots, N\} \begin{cases} \sum_{k=1}^{N} x_{ik} \leq 1 \\ \sum_{k=1}^{N} x_{kj} \leq 1 \\ \sum_{(k,l)\in I(i,j)} x_{kl} \leq 1 \\ \sum_{(k,l)\in J(i,j)} x_{kl} \leq 1 \\ x_{ij} \in \{0, 1\} \end{cases}$$

where $I(i,j)$ and $J(i,j)$ are the sets of squares which a chess queen threatens along two diagonals from the square (i,j), including the square (i,j). In this form we have four groups of constraints: row, column, and two diagonal constraints, one of each type per square. For this problem it is easy to find a subset of constraints that provides the tight upper bound. If we sum up N inequalities corresponding only to row constraints for squares from different rows, we get N as the upper bound: $\sum_{i=1}^{N}\sum_{j=1}^{N} x_{ij} \leq N$. Beginning with $N = 4$, the problem has an N-Queen solution[Hoffman et al., 1969] (see Figure 3).

Though PHP provides the tight upper bound, it is not always immediately clear how to construct an optimal solution that

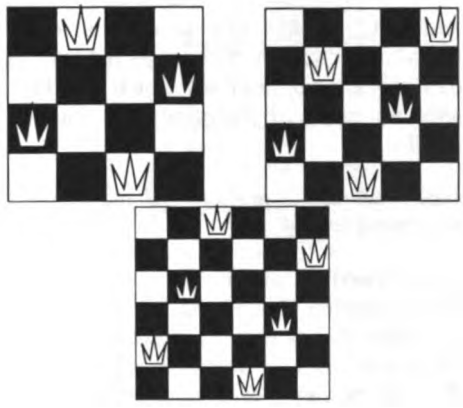


Figure 3: N-Queen Problem Solutions for N = 4, 5, 6

will attain the obtained bound. The *N*-Queen problem stimulated the development of a generation of backtracking algorithms constructing optimal solutions for the *N*-Queen problem, which is a hard problem itself. The Chess Knight problem[Bennett and Potts, 1967] is one of the jewels of PHP applications; it is the one for which the selection of a constraint subset is a challenging task. We discuss this problem in the following section in detail.

3 PHP vs. LPR

In this section, we discuss relations between the Pigeonhole Principle and the Linear Programming Relaxation. For some problems, it is tempting to apply LPR in a brute-force manner. Instead of deriving an additional heuristic that will suggest how to select the desired subset of constraints, one can apply already developed methods of Linear Programming with the hope of deriving the same or even better bound. This brings us back to the main question of this paper: Is it true that PHP and LPR always provide the same upper bound? The following theorem answers this question:

Theorem 1 *If an Integer Programming problem admits a proof by the Pigeonhole Principle, then LPR provides the same optimal value. Conversely, a bound derived by LPR can be matched by PHP.*

Proof: If an IP problem admits PHP in deriving the tight upper bound, then there exists a feasible integer solution x^* attaining the derived bound. On the other hand, there exists a subset of constraints $\hat{J}$, sum of the left-hand sides of which is an integer multiple of the goal function: $\sum_{j\in\hat{J}}\sum_i a_{ij}x_i = k\sum_i c_i x_i$, and the value of x^* is the same multiple of bounding constants: $\sum_{j\in\hat{J}} b_j = k\sum_i c_i x_i^*$.

If we apply LPR to this IP problem, it will provide an integer or fractional optimal solution x^{**} for a relaxed problem. Since PHP has derived the tight upper bound, and the relaxed problem is obtained from the original IP problem by dropping integrality requirements, x^* was one of the candidates for LPR's optimal solution, and the value of LPR's solution x^{**} is equal to PHP's tight upper bound: $\sum_{i=1} c_i x_i^{**} = \sum_{i=1} c_i x_i^*$.

If an optimal integer feasible solution x^* is not known, PHP establishes an upper bound. We show that this bound is equal to the value of the LPR's solution $\sum_{i=1} c_i x_i^{**}$. According to Linear Programming theory, there exists a subset of the constraint set, called "active" constraints, such that the goal function is a positive weighted sum of the left-hand sides of the constraints from this subset $\sum_{j\in\hat{J}}\sum_i \alpha_j a_{ij}x_i = \sum_i c_i x_i$. Moreover, optimal solution x^{**} has no slack for each of the constraints from $\hat{J}$, that is, satisfies them as equality $\sum_i a_{ij}x_i^{**} = b_j$ for $j \in \hat{J}$. Since all the coefficients in the constraints J and the goal function are integer, all the weights $\alpha_j \geq 0$ (positive coefficients) of the weighted sum are rational. We can find integer k to scale rational coefficients up $\beta_j = k\alpha_j$ and make all of β_j integer. Integer k now plays the role of a scaling coefficient, integer β_j tells how many times should the "active" constraint $j \in \hat{J}$ be used in an "unweighted" sum of left-hand sides $\sum_{j\in\hat{J}}\beta_j\sum_i a_{ij}x_i = k\sum_i c_i x_i$. Hence, the upper bound $1/k\sum_{j\in\hat{J}}\beta_j b_j = \sum_i c_i x^{**}$ can be matched by PHP.

Thus, the set of "active" constraints forms the desired subset $\hat{J}$ and positive integer weights determine the number of repetitions in $\hat{J}$. Therefore, the value of the solution derived by LPR does not improve the upper bound provided by PHP and, conversely, it can be mimicked by PHP to establish the same upper bound. ■

Theorem 1 provides the following answer to the main question of the paper: *If the optimal value is obtained by PHP for a combinatorial optimization problem stated in the Integer Programming form, LPR provides the same bound as PHP.* We demonstrate that this result is nontrivial by solving the Chess Knight problem[Bennett and Potts, 1967]: "What is the maximal number of chess knights that can be placed on the 8x8 chessboard in such a way that they do not attack each other?"

The classical elegant solution relies on the existence of a Hamiltonian tour of length 64 following the knight moves. One of such tours is presented in Figure 4.

One can split the tour into 32 pairs of chess squares that are adjacent in the sense of the knight move, and apply PHP: Each pair can contain at most one knight, otherwise two knights would attack each other. For example, consider pairs of squares (1, 2), (3, 4),...,(63, 64). None of them can accommodate more than one knight (see Figures 4 and 5). Thus, 32 pairs of adjacent squares can accommodate at most 32 knights. Although this simple proof does not provide us with a solution to the problem, (for example, it allows to place knights on squares 4 and 5), it gives an upper bound.

Moreover, this proof gives an impression of using a "hidden" application of the Pigeonhole Principle, whereas the Hamiltonian tour is just a heuristic for finding a subset of the constraint set. The Chess Knight problem for the stan-

8	43	24	53	6	41	22	51
25	54	7	42	23	52	5	40
44	9	62	15	46	31	50	21
55	26	45	32	63	14	39	4
10	33	16	61	30	47	20	49
27	56	29	64	13	60	3	38
34	11	58	17	36	1	48	19
57	28	35	12	59	18	37	2

Figure 4: Hamiltonian Tour on a Chessboard for the Knight

dard chessboard can be presented as the following Integer Programming problem:

$$max(\sum_{i=1}^{8}\sum_{j=1}^{8} x_{ij})$$

$$x_{ij} + x_{kl} \leq 1 \quad i = 1,\ldots,8 \;\; j = 1,\ldots,8 \;\; (k,l) \in U(i,j)$$

$$x_{ij} \in \{0,1\}$$

where x_{ij} represents the amount of knights in the square (i,j); $U(i,j)$ is the set of squares on the chessboard which a chess knight threatens from the cell (i,j) (see Figure 5).

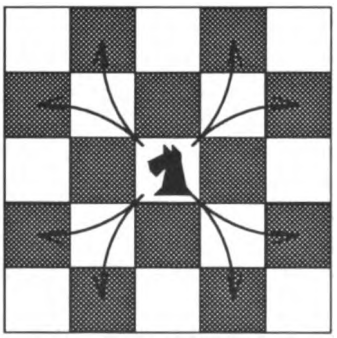


Figure 5: The Set of Adjacent Squares for the Knight on a Chessboard

If applied to the Chess Knight problem, the Linear Programming Relaxation method provides the same upper bound of 32. Unlike PHP, LPR does not require heuristics to identify any subset of constrains, its calculational routine considers "active" constraints inside the solving process. However, LPR is likely to produce fractional solutions, say $x_i = 1/2$ for $i = 1,2,\ldots,64$. Theorem 1 shows that, if applied, the original Pigeonhole Principle will provide the same value. So, if the Hamiltonian tour heuristic were not known, the application of LPR would tell that 32 is the best upper bound that can be obtained by the original PHP. Since a chess knight alternate colors (see Figure 5), one can place 32 knights on the chess squares of the same color, thus attaining the optimal value and constructing the optimal solution for the Chess Knight problem.

We showed that, if an optimal solution of value B for a combinatorial optimization problem is derived by the Pigeonhole Principle, an optimal solution for the last problem in the following chain of representation changes has the same value B:

> Original combinatorial problem→ IP problem → Linear Programming problem.

Examples discussed in this section showed that an additional effort is needed to apply the Pigeonhole Principle. For some problems it is easy to match problem's objects with pigeons and holes, in some cases it is a state of art. Often PHP applications hint on how to construct optimal solutions, though for some problems it is an independent difficult problem.

In its turn, LPR can be applied to any instance of an IP problem without need in additional knowledge. Unfortunately, LPR often outputs a fractional optimal solution, which does not shed any light on how to transform it into an integer one of the same value. We continue the discussion on benefits and disadvantages of PHP and LPR in the next section.

4 APPLICATIONS OF PHP THAT REQUIRE REPRESENTATION CHANGES

In the previous sections, we were able to apply PHP in a brute-force manner, because each Integer Programming problem contained a subset of constraints, sum of which provided the desired bounds for values of the goal functions. We call such a case a regular application of PHP, as opposed to the "Hidden" Pigeonhole Principle (HPHP) that requires additional representation changes and heuristic knowledge to fulfill a similar task.

This section is devoted to the discussion on HPHP and its relation to the original Pigeonhole Principle. To make it more intuitive, we illustrate the discussion by the classical Mutilated Checkerboard [Newell, 1965] and the Firm Tiling problems:

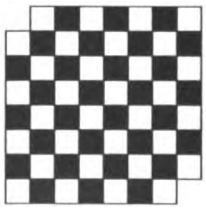


Figure 6: A Mutilated Checkerboard

Mutilated Checkerboard Problem: Consider an NxN checkerboard with two opposite corners removed (see Figure 6). Can one cover this "mutilated" checkerboard completely by non-overlapping domino pieces, each of the size of two squares of the checkerboard?

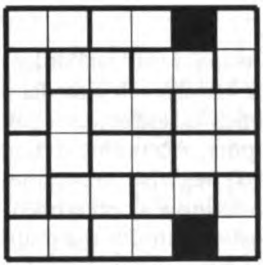


Figure 7: Firm 17-Tile Solution.

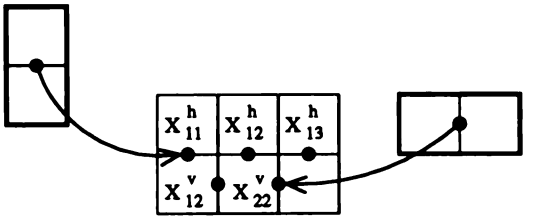


Figure 8: Modeling Domino Overlap as a Set of Inequalities

Firm Tiling Problem: Consider a checkerboard of size 6x6 made of a soft square cloth and 18 hard tiles of size 1x2. Can one glue all 18 tiles to such a checkerboard, so that the "middle-cut" requirement is satisfied, i.e. each splitting line inside the checkerboard goes through the middle line of at least one tile?

Figure 7 shows a firm 17-tile solution. The "middle-cut" requirement for a particular splitting line restricts the cloth to be folded along this splitting line: If the line crosses the middle of at least one tile, the cloth cannot be folded along this line unless the tile is broken. This is what we call a *"single-firm"* tiling. A more restrictive *"double-firm"* tiling requires each splitting line to cross middle lines of at least two tiles. In this section we show that a *"single-firm"* complete tiling implies a *"double-firm"* tiling in the Firm Tiling problem.

Following the proposed approach, the first step needed is the representation change converting the above problems into Integer Programming problems. To accomplish this step, one needs to model the fact that domino pieces do not overlap in the Integer Programming form. We assign variables x^v_{ij} to vertical splits, where i is the row number and j is the split in ith row between squares (i,j) and $(i,j+1)$. In the same way, we define variables x^h_{ij} for horizontal splits between squares (i,j) and $(i+1,j)$. One-valued variables $x^v_{ij} = 1$ model the horizontal placement of a domino piece (tile) in such a way so that its middle line is located at ith vertical splitting line and jth row; one-valued variables $x^h_{kl} = 1$ model the vertical placement of a domino piece (tile) with its middle line at kth horizontal split in lth column. In these terms, non-overlapping can be represented by the following set of constraints:

$$x^v_{ij} + x^v_{i-1j} \le 1 \quad x^v_{ij} + x^v_{i+1j} \le 1 \quad x^v_{ij} + x^h_{i-1j-1} \le 1 \qquad (9)$$

$$x^v_{ij} + x^h_{i-1j} \le 1 \quad x^v_{ij} + x^h_{ij-1} \le 1 \quad x^v_{ij} + x^h_{ij} \le 1 \qquad (10)$$

Figure 8 demonstrates a part of a checkerboard and the correspondence of domino pieces (tiles) placings to 0/1-variables and splits.

In the Mutilated Checkerboard problem the goal function is the unweighted sum of all domino-piece variables: $\sum_{ij} x^v_{ij} + \sum_{ij} x^h_{ij}$. It is easy to guess one of the fractional optimal solutions produced by LPR: $x^*_{**} = 1/2$. It tells that if applied, PHP will provide the same bound of 31 (which is not tight). Furthermore, an optimal fractional solution does not help to find an optimal integer one, even when the upper bound is tight.

If the original checkerboard is of size NxN with N odd, a simple PHP application shows that one can use at most $\frac{N^2-3}{2}$ domino pieces for a non-overlapping covering of the mutilated checkerboard: Each piece contains two squares, and there are $N^2 - 2$ squares in the mutilated checkerboard. Therefore, one can put at most $\lfloor \frac{N^2-2}{2} \rfloor = \frac{N^2-3}{2}$ non-overlapping domino pieces. Actual attempts to cover the mutilated checkerboard with odd sizes readily give an optimal solution of the above value.

If N is even, the original PHP does not put any additional restrictions on the number of pieces. However, none of the attempts to cover the mutilated checkerboard by domino pieces achieves the desired bound of $\frac{N^2-2}{2}$, all constructed solutions provide at most $\frac{N^2-4}{2}$ pieces. Neither PHP nor LRP provide a tight upper bound for even-sized mutilated checkerboards. Nonetheless, the heuristic of two-coloring the mutilated checkerboard and applying PHP to the monochrome set of squares of smaller size completes the proof of the fact that $\frac{N^2-4}{2}$ is actually the optimal value for covering the mutilated checkerboard with even sizes. If we color the original checkerboard in usual black-and-white chess colors (see Figure 6) and then cut off 2 opposite corner squares (of the same color), the remaining mutilated checkerboard contains unevenly colored square sets. The checkerboard presented in Figure 6, for example, has 30 black squares and 32 white squares. Since each domino piece contains 2 squares of the opposite colors, applying PHP to a smaller set of black or white squares, we obtain the tight upper bound of $\frac{N^2-4}{2}$. This argument completes the proof of the Mutilated Checkerboard problem by HPHP. Such a modification of PHP is favorable in comparison with LPR and the original PHP, because it is capable of deriving the optimal value of the goal function. As we mentioned before, Linear Programming Relaxation provides $\frac{N^2-2}{2}$ as an upper bound. According to Theorem 1, if applied, PHP outputs exactly the same upper bound.

The idea of two-coloring is a simple elegant heuristic that allows to apply the Hidden Pigeonhole Principle and derive the tight upper bound. After being known for several dozen years, this application of HPHP might seem to be too

simple to stimulate the development of alternative methods. We introduce the Firm Tiling problem as an example of a problem which is hard to solve without prior knowledge of the appropriate heuristic. For example, multi-coloring does not help to find the desirable matching between tiles, checkerboard squares, pigeons and holes.

Theory of Integer Programming suggests to apply Integer cuts, for example, Chvatal-Gomory's cuts, to extend the set of valid inequalities. The main idea of the Chvatal-Gomory's cut (CG-cut) is to use the integrality of variables in feasible solutions. If a weighted sum of the inequalities derived so far contains the left-hand side with integer coefficients, the right-hand side can be rounded down to the nearest integer: $\sum_{i=1}^{N} a_i x_i \leq b$ implies $\sum_{i=1}^{N} a_i x_i \leq \lfloor b \rfloor$, for integer a_i and x_i $i = 1, \ldots, N$. In a certain sense CG-cuts look as simple as the original Pigeonhole Principle. However, CG-cuts allow to derive constraints that cannot be obtained from the initial set of inequalities by taking weighted positive sums.

We, first, demonstrate the correctness of the Integer Programming formulation of the Mutilated Checkerboard problem and then derive an upper bound for it by means of Chvatal-Gomory's cuts and the Hidden Pigeonhole Principle.

Since we are about to apply some of the techniques of Integer cuts to the Mutilated Checkerboard problem, we first demonstrate simple reductions from Integer Programming theory. For example, Lemma 1 builds a reduction from the set of pairwise constraints with 0/1-vertex variables to the Exclusive Rule for a clique K_N, where a clique is a complete graph with $N \geq 2$ vertices.

Lemma 1 *If a clique K_N admits exclusively 0/1-assignments to its vertices $v_1, \ldots, v_N$ and, for each pair of vertices (v_i, v_j), at most one vertex can be assigned to 1, then there is at most one vertex assigned to 1 in the whole clique K_N.*

Proof: There are $\frac{N(N-1)}{2}$ inequalities of the type $x_i + x_j \leq 1$. We would like to prove that this collection of constraints implies a single clique inequality $\sum_{i=1}^{N} x_i \leq 1$ (Exclusive Rule) for 0/1-variables $x_i \in \{0, 1\}$ $i = 1, \ldots, N$. We prove it by induction on the number of vertices.
Induction Base: If $N = 2$, the given inequality and the clique inequality $x_1 + x_2 \leq 1$ coincide.
Induction Step: Suppose that we can derive all N clique inequalities for each of the N sub-cliques of size $N - 1$. If we sum them up, we get:

$$\sum_{j=1}^{N} \sum_{i \neq j} x_i \leq \sum_{j=1}^{N} 1,$$

which implies that $(N-1)\sum x_i \leq N$ or, equivalently, $\sum x_i \leq N/(N-1)$. Since $N > 2$, $\lfloor \frac{N}{N-1} \rfloor = \lfloor 1 + \frac{1}{N-1} \rfloor = 1$. If we apply CG-cut to the last inequality, we get the desired N-clique inequality:

$$\sum_{i=1}^{N} x_i \leq 1. \blacksquare \qquad (11)$$

Four (or less) splits that form the border of the square (i,j) correspond to four variables that model placing of domino pieces (tiles). Corner or side squares border with only two or three internal splits. According to Lemma 1, if we consider all six (or less) pairwise inequalities (9-10) involving four variables bordering a checkerboard square, CG-cuts provide the Exclusive Rule for the clique associated with the square. This Exclusive Rule corresponds to the requirement of non-overlapping of domino pieces over the square (i,j). The addition of the clique inequalities derived by CG-cuts to the initial set of constraints constitutes a representation change of the IP problem.

Lemma 2 *If clique inequalities for all squares of the NxN mutilated checkerboard (N is even) are added to the set of constraints, the optimal value of the relaxed Linear Programming problem is $\frac{N^2-4}{2}$.*

Proof: Consider the following subset of clique constraints: Pick a monochrome subset of squares of smaller size and sum up all the clique constraints corresponding to these squares (of the same color). Each clique constraint is an unweighted sum of four (or less) clique variables $x_{i_1} + x_{i_2} + x_{i_3} + x_{i_4} \leq 1$. Since squares of the same color do not share sides, the sum of all clique constraints corresponding to squares of the same color is an unweighted sum of variables. On the other hand, all variables are presented in the final sum, because a legitimate placement of a domino piece covers squares of both colors. Since the number of clique constraints corresponds to the number of monochrome squares of smaller size, the sum of the constraints establishes a tighter bound for the goal function $\sum x_i \leq \frac{N^2-4}{2}$. Knowing this bound, it is easy to come up with a solution for an even-sized Mutilated Checkerboard problem that attains this value. ■

Thus, the proof that uses HPHP, relies on additional knowledge that each domino piece covers a bi-chromatic configuration, whereas the combination of LPR with CG-cuts takes into account this argument automatically. CG-cuts expand the set of constraints by adding clique inequalities for all checkerboard squares. After that LPR solves the new Integer Programming problem. We identify the expansion of the constraint set with the representation changes for the original Integer Programming problem. If the two-coloring heuristic were not known, then LPR with CG-cuts could be used to obtain the tight upper bound of $\frac{N^2-4}{2}$. After that, it is relatively easy to construct a solution attaining this value, which is proven to be optimal. The relations between HPHP and LPR with Integer cuts in solving the Mutilated Checkerboard problem are very similar to those between PHP and LPR.

The Firm Tiling problem does not admit the proof by the original PHP, because an obvious fractional solution $x^*_{**} = 1/2$ is feasible for a relaxed LP problem and the "middle-cut" requirement

$$\sum_{j=1}^{6} x^h_{ij} = \sum_{j=1}^{6} \tfrac{1}{2} = 3 \geq 1 \quad i = 1, 2, \ldots, 5$$

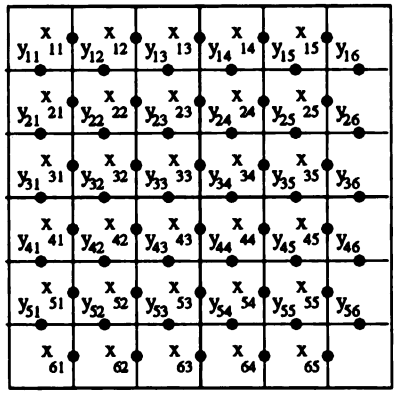


Figure 9: Modeling a Firm Tiling of a 6x6 Checkerboard.

$$\sum_{i=1}^{6} x_{ij}^{v} = \sum_{i=1}^{6} \tfrac{1}{2} = 3 \geq 1 \quad j = 1, 2, \ldots, 5$$

is satisfied for all internal splitting lines. However, brute-force attempts to construct a complete firm tiling fail to provide an optimal 18-tile solution. The best firm tiling consist of at most 17 non-overlapping tiles, Figure 7 shows one of such tilings.

Nonetheless, the Hidden Pigeonhole Principle is capable of setting the tight upper bound of 17 tiles for this problem. To preserve the beauty of the elegant solution by HPHP for now, we first establish the tight upper bound through the combination of LPR and CG-cuts. We show that if one more constraint

$$\sum x_{ij}^{h} + \sum x_{ij}^{v} > 17 \tag{12}$$

is added to the original IP problem, the set of feasible integer solutions becomes empty.

Figure 9 presents 6x6 checkerboard with 0/1-variables assigned to its internal splits. To avoid superscript notations we denote x_{ij}^{v} as x_{ij} and x_{ij}^{h} as y_{ij}. Since x_{ij} and y_{ij} are integer, inequality (12) implies

$$\sum x_{ij} + \sum y_{ij} \geq 18. \tag{13}$$

In its turn, the latter inequality implies that the whole checkerboard should be covered by tiles. Since clique inequalities prohibit tiles from overlapping, each square is covered exactly by half-a-tile[2]. This is not a surprising conclusion, as we are attempting to cover a 6x6 checkerboard by 18 non-overlapping tiles.

Lemma 3 *Inequality (13) and clique inequalities (11) for all squares of the 6x6 checkerboard imply* "double-firm" *tiling.*

Proof: We prove the statement of the Lemma by induction. In the induction base, we show it for the leftmost vertical splitting line. Due to the symmetry this proof remains unchanged for horizontal splitting lines.

Induction Base: If we consider the leftmost column of a checkerboard presented in Figure 9, inequalities (9,10) and (13) imply the following six equalities:

$$x_{11} + y_{11} = 1 \quad x_{21} + y_{11} + y_{21} = 1 \quad x_{31} + y_{21} + y_{31} = 1 \tag{14}$$

$$x_{61} + y_{51} = 1 \quad x_{41} + y_{31} + y_{41} = 1 \quad x_{51} + y_{41} + y_{51} = 1 \tag{15}$$

The "middle-cut" requirement for the leftmost vertical splitting line corresponds to the following inequality:

$$\sum_{i=1}^{6} x_{i1} \geq 1 \tag{16}$$

Sum of equalities (14) and (15) produces a new constraint:

$$\sum_{i=1}^{6} x_{i1} + 2\sum_{j=1}^{5} y_{j1} = 6 \tag{17}$$

Now we can subtract (16) from (17) and divide it by two:

$$2\sum_{j=1}^{5} y_{j1} \leq 5 \quad \textit{implies} \quad \sum_{j=1}^{5} y_{j1} \leq 2.5 \tag{18}$$

Since all y_{j1} in the left-hand side of inequality (18) are integer variables, we can apply CG-cut to obtain a new (tighter) inequality:

$$\sum_{j=1}^{5} y_{j1} \leq 2 \tag{19}$$

In its turn, constraints (17) and (19) imply

$$\sum_{i=1}^{6} x_{i1} \geq 2 \tag{20}$$

Induction Step: Suppose that we have proved that inequalities (11) and (13) imply the *"double-firm"* tiling for $j_0 - 1$ leftmost columns of the checkerboard ($1 \leq j_0 - 1 \leq 4$). It implies that

$$\sum_{i=1}^{6} x_{ij} \geq 2 \quad j = 1, \ldots, j_0 - 1 \tag{21}$$

Consider the tiling of left j_0 columns. All tiles $y_{ij} = 1$ with $i \leq j_0$ cover two checkerboard squares in left j_0 columns, the same is true for tiles $x_{ij} = 1$ with $i < j_0$. These tiles contribute two to the amount of covered squares in the discussed portion of the checkerboard. Tiles $x_{ij_0} = 1$ contribute one to the number of tiled squares in left j_0 columns. If we count the number of tiled squares in left j_0 columns according to the above observation and take into account that each square should be covered by a tile, we get the following equation:

$$\sum_{j=1}^{j_0}\sum_{i=1}^{5} 2y_{ij} + \sum_{j=1}^{j_0-1}\sum_{i=1}^{6} 2x_{ij} + \sum_{i=1}^{6} x_{ij_0} = 6j_0 \tag{22}$$

Since a *"single-firm"* tiling requires $\sum_{i=1}^{6} x_{ij_0} \geq 1$, equality (22) and "firmness" imply

$$\sum_{j=1}^{j_0}\sum_{i=1}^{5} 2y_{ij} + \sum_{j=1}^{j_0-1}\sum_{i=1}^{6} 2x_{ij} \leq 6j_0 - 1 \tag{23}$$

[2]This fact can be proved by considering a subset of clique constraints corresponding to a monochrome set of checkerboard squares in the way similar to the Mutilated Checkerboard problem.

Inequality (23) can be divided by two:

$$\sum_{j=1}^{j_0}\sum_{i=1}^{5} y_{ij} + \sum_{j=1}^{j_0-1}\sum_{i=1}^{6} x_{ij} \leq 3j_0 - 1/2 \quad (24)$$

If we apply CG-cut to inequality (24), we get a tighter constraint:

$$\sum_{j=1}^{j_0}\sum_{i=1}^{5} y_{ij} + \sum_{j=1}^{j_0-1}\sum_{i=1}^{6} x_{ij} \leq 3j_0 - 1 \quad (25)$$

which together with equality (22) imply the *"double-firm"* tiling of j_0th vertical splitting line:

$$\sum_{i=1}^{6} x_{ij_0} \geq 2 \quad (26)$$

■

Lemma 3 constitutes the hardest part of the Firm Tiling problem. Were we given the *"double-firm"* tiling requirement as the part of the initial problem, a simple application of the Pigeonhole Principle would provide the negative (infeasible) answer: Since there are ten splitting lines, each passing the middle line of at least two tiles, and none of the tiles can be shared by splitting lines in such counting, one needs at least 20 tiles for a *"double-firm"* tiling, in which case they would overlap. The Integer cuts technique demonstrated in Lemma 3 proves infeasibility through deriving the *"double-firm"* tiling requirement from a required *"single-firmness"* and the completeness of tiling (13). Since inequality (13) is actually an equality for the 6x6 checkerboard, the *"double-firm"* tiling inequalities make the Firm Tiling problem infeasible.

From a glance, Integer cuts seem to be manipulating with halfs and other fractionals in a beneficial manner. However, it is not just a game with fractionals. Integer cuts perform methodological "squeezing" of the polygon of feasible solutions remaining all integer solutions feasible. Moreover, new constraints obtained through Integer cuts can not be derived by taking positive weighted sums of the existing constraints.

Although both problems presented in this section do not admit proofs by the original Pigeonhole Principle, it is possible to perform a representation change of the problem statements and transform both problems into ones that admit proofs by the Pigeonhole Principle. We call such method of solving Integer Programming problems as the Hidden Pigeonhole Principle (HPHP).

Such application of HPHP are useful mainly for deriving the tight upper bound. However, for some problems precise knowledge of the optimal value allows AI planning systems to construct an optimal solution. For the problems presented in this section this can be done, for example, by placing domino pieces randomly or greedily and applying the backtracking techniques when necessary. Success in constructing an optimal solution from the optimal value depends heavily on planning domain properties. Nonetheless, without a HPHP application the optimal value would be unknown, and any attempts to construct a solution attaining non-tight upper bound would fail.

5 CONCLUSIONS

In this paper, we showed how through an effective representation change we solved the problem of comparing the bounding power of the Pigeonhole Principle and the Linear Programming Relaxation method. We proved that both methods establish the same upper bounds, when applied to the same problems. The Pigeonhole Principle is more intuitive and often shows whether the upper bound is tight, whereas Linear Programming Relaxation can be applied to any instance of an Integer Programming problem without the use of additional knowledge or representation changes.

We illustrated our new representation change in several combinatorial optimization problems, and demonstrated how this representation change can be utilized to provide an alternative way of solving such problems, which is easier to implement in AI planning systems than PHP or HPHP.

Acknowledgements

This research is sponsored in part by the National Science Foundation under grant number IRI-9502548. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations or the U.S. government.

Thanks to Bart Selman and Matthew Ginsberg for fruitful discussions, to Eugene Fink for creative comments.

References

[Bennett and Potts, 1967] Bennett, B. and Potts, R. (1967). Arrays and brooks. *Journal for the Australian Mathematical Society*, pages 23–31.

[Ginsberg, 1996] Ginsberg, M. (1996). Personal Communications.

[Hoffman et al., 1969] Hoffman, E., Loessi, J., and Moore, R. (1969). Constructions for the solution of the *m* queens problem. *National Mathematics Magazine*, March-April:66–72.

[Khachian, 1979] Khachian, L. (1979). A polynomial algorithm in linear programming. *Soviet Math. Doklady*, 20(1).

[Nauck, 1850] Nauck (1850). Schach. *Illustrierte Zeitung*, 361:352.

[Nemhauser and Wolsey, 1985] Nemhauser, G. and Wolsey, L. (1985). *Integer and Combinatorial Optimization*. John Wiley & Sons.

[Newell, 1965] Newell, A. (1965). Limitations of the current stock of ideas about problem solving. In Kent, A. and Tualbee, O., editors, *Electronic Information Handling*. Spartan Books, Washington, DC.

[Thucker, 1980] Thucker, A. (1980). *Applied Combinatorics*. John Wiley & Sons.

On the role of Disjunctive Representations and Constraint Propagation in Refinement Planning

Subbarao Kambhampati and Xiuping Yang
Department of Computer Science and Engineering
Arizona State University, Tempe, AZ 85287.
Email: { rao, xiuping.yang }@asu.edu
WWW: http://rakaposhi.eas.asu.edu/yochan.html

Abstract

Most existing planners intertwine the refinement of a partial plan with search by pushing the individual refinements of a plan into different search branches. Although this approach reduces the cost of handling partial plans, it also often leads to search space explosion. In this paper, we consider the possibility of handling the refinements of a partial plan together (without splitting them into search space). This is facilitated by disjunctive partial plan representations that can compactly represent large sets of partial plans. Disjunctive representations have hitherto been shunned since they may increase the plan handling costs. We argue that performance improvements can be obtained despite these costs by the use of (a) constraint propagation techniques to simplify the disjunctive plans and (b) CSP/SAT techniques to extract solutions from them. We will support this view by showing that some recent promising refinement planners, such as the GRAPHPLAN algorithm [2], can be seen as deriving their power from disjunctive plan representations. We will also present a new planning algorithm, UCPOP-D, which uses disjunctive representations over UCPOP [19] to improve performance. Finally, we will discuss the issues and tradeoffs involved in planning with disjunctive representations.

1 Introduction

A large part of the work on plan synthesis in artificial intelligence falls under the rubric of refinement planning. Refinement planning [12] involves manipulating sets of partial plans, each of which are shorthand notations for a set of potential solutions for the planning problem (called the candidate set of the plan). The planning process starts with the null plan, corresponding to the set of all action sequences, and consists of two repeated steps. First, called the solution-extraction phase, involves examining current set of partial plans to see if they contain a solution. If this step fails, then a "refinement strategy" is applied to the current set of plans to get a new plan set. Informally, refinement strategies can be understood as operations that narrow the candidate set by eliminating those sequences that cannot be solutions to the problem.

As described above, refinement planning does not need any explicit search. However, most refinement planners described in the literature, including the popular ones like UCPOP [19] and PRODIGY [4] introduce explicit search into the refinement process by considering each of the refinements of a plan in a different search branch. The motivation behind introducing search into refinement planning is to restrict the solution extraction and refinement operations to single plans, thereby making them cheaper. The expense that all these planners pay for this reduction in "per-plan" cost is the increase in search space size. Indeed, it is well known that planners such as UCPOP and PRODIGY generate very large search spaces even for simple problems [7, 2]. The usual solution to this problem is to control the planner's search with the help of search control knowledge acquired from domain experts (e.g. task reduction schemas) or through learning techniques (e.g. explanation-based learning [15], case-based planning [6]).

In this paper, we will consider a more direct solution to the search space explosion problem – that of handling sets of plans without splitting them into the search space. At first glance, this seems to involve a mere exchange of complexity from search space size to solution extraction cost. In particular, handling sets of plans together might lead to unwieldy data structures, as well as a costly solution-extraction process. We will argue that we can nevertheless derive performance improvements by the use of disjunctive partial plan representations, which support compact representation of large sets of partial plans, constraint propagation techniques which simplify the partial plan constraints, and the use of efficient CSP/SAT solvers to help in solution extraction.

The paper is organized as follows. Section 2 provides the background on the planning problem, and syntax and semantics of partial plans, which can be skipped by read-

ers familiar with candidate set semantics for partial plans (c.f. [12]). The next two sections provide a novel view of the notions of refinement strategies and refinement planning which shows the secondary nature of search in refinement planning. Section 5 discusses how disjunctive representations and constraint propagation can be used to efficiently handle sets of plans together, without splitting them into the search space. It also explains the success of GRAPHPLAN algorithm in terms of these ideas. Section 6 shows how these ideas can be used to improve the performance of traditional refinement planners by presenting UCPOP-D algorithm which uses disjunctive representations to improve the performance of UCPOP. This section also describes empirical results demonstrating the potential of UCPOP-D, and the ways in which it can be extended. Section 7 discusses the tradeoffs in planning with disjunctive representations and Section 8 discusses the relations with other recent efforts to scale up AI planning techniques. Section 9 summarizes the contributions of this paper.

2 Preliminaries

A planning problem is defined in terms of the initial and goal states of the world, and a set of actions. The world states are represented in terms of some binary state variables, and the actions transform a given state into another. Actions are described in terms of preconditions (i.e., the specific state variable/value configuration that must hold in the world state for them to be applicable) and effects (i.e., the changes that the action would make to state variable values to give rise to the new world state). A solution to the planning problem is a sequence of actions that when executed from the initial state, results in a state where the state variables mentioned in the goal state have the specified values.

As an example, consider the simple one-way rocket domain, consisting of a single rocket, and two packages, A and B, all of which are initially on earth. The objective is to send both the packages to moon. There are three actions, $Load(x)$ which puts the package x in the rocket, $Unload(x)$ which takes the package out of the rocket (and on to earth or moon, whereever the rocket may be at that time). Finally, there is the $Fly()$ action which transports the rocket, along with its contents, from earth to moon. The preconditions and effects of $Fly()$ action are specified as follows:

Fly()
Preconditions: $At(R, E)$
Effects: $At(R, M)$
$\forall_x In(x) \Rightarrow [At(x, M), \neg At(x, E)]$

The initial state is specified as $At(A, E) \wedge At(B, E) \wedge At(R, E)$ and the goal state is specified by $At(A, M) \wedge At(B, M) \wedge \neg In(A) \wedge \neg In(B)$.

Refinement planners [12] attempt to solve a planning problem by navigating the space of *sets of potential solutions (action sequences)*. The potential solution sets are represented and manipulated in the form of "(partial) plans."

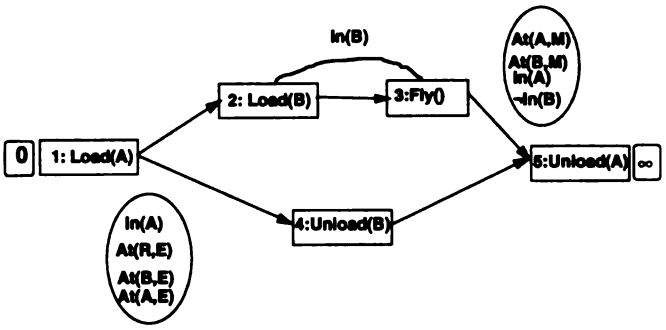


Figure 1: This figure depicts the partial plan from the rocket domain. The ordering constraints between steps are shown by arrows. The interval preservation constraints are shown by arcs. Contiguous steps are shown immediately next to each other. States of the world after the prefix and before the suffix of the partial plan are shown in the ovals beside them.

Syntactically, a partial plan $\mathcal{P}$ can be seen as a set of constraints. Semantically, a partial plan is a shorthand notation for the set of action sequences that are consistent with its constraints. The set of such action sequences is called the set of candidates (or the **candidate set**) of the partial plan $\mathcal{P}$ and is denoted as $\langle\langle \mathcal{P} \rangle\rangle$. Informally, refinements narrow the candidate sets of partial plans by gradually eliminating action sequences that cannot be solutions to the problem. To make this more precise, we need to talk about the syntax and semantics of the partial plans.

2.1 Partial Plans: Syntax

The following representation of partial plans is a generalization of representations used in several existing planning algorithms, and shows the types of constraints normally used.[1] A *partial plan* consists of a set of *steps*, a set of *ordering constraints* that restrict the order in which steps are to be executed, and a set of *auxiliary constraints* that restrict the value of state variables (describing the states of the world) over particular points or intervals of time. Each step is associated with an action. To distinguish between multiple instances of the same action appearing in a plan, we assign to each step a unique step number i and represent the ith step as the pair $i : A_i$ where A_i is the action associated with the ith step. The step 0 corresponds to a dummy action symbolizing the beginning of the plan, and the step ∞ corresponds to the dummy action symbolizing the end of the plan. The conditions true in the initial state are considered the effects of 0 and the conditions needed in the goal state are considered the preconditions of ∞. Figure 1 shows a partial plan $\mathcal{P}_{eg}$ consisting of seven steps (including 0 and ∞). The plan $\mathcal{P}_{eg}$ is represented as follows:

[1] For a very different partial plan representation, that still has candidate set based semantics, see Ginsberg's paper in these proceedings [5].

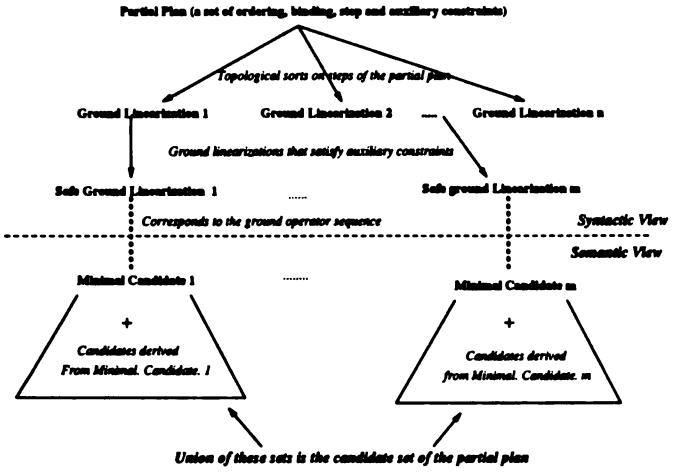


Figure 2: Relations between a partial plan and its candidate set.

$$\left\langle \begin{array}{c} \{(1:Load(A)),(2:Load(B)),(3:Fly()), \\ (4:Unload(B)),(5:Unload(A)),\}, \\ \{(0*1),(1\prec 2),(1\prec 4),(2\prec 3), \\ (3\prec 5),(4\prec 5),(5*\infty)\}, \\ \{(2 \overset{In(B)}{-} 3)\} \end{array} \right\rangle$$

An ordering constraint of the form $(i \prec j)$ indicates that Step i precedes Step j. An ordering constraint of the form $(i * j)$ indicates that Step i is contiguous with Step j, that is Step i precedes Step j and no other steps intervene. When steps are contiguous to 0, we can completely predict the state of the world after those steps. In the example plan, the state of the world after step 1 is shown in the oval below the plan. Similarly, we can predict the minimal requirements on the state of the world before a step that is contiguous to ∞.

An auxiliary constraint of the form $(i \overset{P}{-} j)$ is called an *interval preservation constraint* and indicates that P is to be preserved in the range between Steps i and j (and therefore no operator with postcondition $\neg P$ should occur between Steps i and j). In particular, according to the constraint $(2 \overset{In(B)}{-} 3)$, Step 4 should not occur between Steps 2 and 3. An auxiliary constraint of the form $P@s$ is called the *point truth constraint* (PTC), and requires that the condition P be true in the state in which s is executed. A partial plan containing a step $s : A$ will have PTCs corresponding to all the preconditions of A. In addition, there may be PTCs corresponding to secondary preconditions of A posted to make A either preserve or cause some required condition.

2.2 Partial Plans: Semantics

The semantic status of a partial plan constraint is clarified by specifying when a given action sequence is said to satisfy the constraint. In particular, an action sequence is a candidate of a partial plan if it contains actions corresponding to all the steps of the plan, in an order consistent with the precedence and contiguity constraints, and it satisfies all the interval preservation constraints [12]. Figure 2 shows the schematic relations between a partial plan and its candidate set [12], and we will illustrate these relations with respect to the example plan in Figure 1. Each partial plan corresponds to a set of topological sorts (e.g. $\langle 1,2,3,4,5\rangle$ and $\langle 1,2,4,3,5\rangle$). The subset of these that satisfy the auxiliary constraints of the plan (e.g. $\langle 1,2,3,4,5\rangle$) are said to be the safe-ground linearizations of the plan. ($\langle 1,2,4,3,5\rangle$ is not a safe linearization since $4 : Unload(B)$ violates the interval preservation constraint $(2 \overset{In(B)}{-} 3)$). Each safe ground linearization of the plan corresponds to an action sequence which is a minimal candidate of the partial plan (e.g. $\langle Load(A), Load(B), Fly(), Unload(B), Unload(A)\rangle$). An infinite number of additional candidates can be derived from each minimal candidate of the plan by augmenting (padding) it with additional actions without violating the auxiliary constraints (e.g. $Load(A), Unload(A), Load(A), Load(B), Fly(), Unload(B), Unload(A)$).

Thus, the candidate set of a partial plan is infinite, but the set of its minimal candidates is finite. The solution extraction involves searching the minimal candidates of the plan to see if any of them are solutions to the planning problem. The process of refinement can be understood as incrementally increasing the size of these minimal candidates so that action sequences of increasing lengths are examined to see if they are solutions to the problem.

3 Refinement Strategies

Refinement strategies narrow the candidate sets of partial plans by adding constraints to eliminate action sequences that cannot be solutions. They differ based on the types of constraints they add (and thus the type of action sequences they eliminate). They are best seen as mapping a set of partial plans to another set of partial plans. Since partial plans represent sets of action sequences, all set theoretic operations – union, intersection etc. – are well defined for them. In particular we will define a **planset** $\widehat{\mathcal{P}}$ to be a set of one or more partial plans $\{\mathcal{P}_1, \mathcal{P}_2 \ldots \mathcal{P}_n\}$. The P_i are called the components of $\widehat{\mathcal{P}}$. The candidate set of $\widehat{\mathcal{P}}$ is defined as the union of the candidate sets of the partial plans in that plan set (a single plan can be seen as a singleton planset). Thus

$$\langle\!\langle \widehat{\mathcal{P}} \rangle\!\rangle \triangleq \langle\!\langle \mathcal{P}_1 \rangle\!\rangle \cup \langle\!\langle \mathcal{P}_2 \rangle\!\rangle \cup \ldots \cup \langle\!\langle \mathcal{P}_n \rangle\!\rangle$$

A planset is said to be *irredundant* if the candidate sets of component plans are all disjoint.[2] We will also denote the set of all action sequences that can be executed from the initial

[2]The differentiation between plans and plansets is an artificial one made for convenience. A planset in one plan language may be a single plan in another plan language. In fact, we shall see below that a set of partial plans can be represented compactly as a single plan containing disjunctive step, ordering and binding constraints.

state I and satisfy the goals G as $\mathcal{L}(I,G)$. The objective of a planning algorithm is to be able to return some desired element of $\mathcal{L}(I,G)$.

A refinement strategy $\mathcal{R}$ is an operation which takes a planset $\widehat{\mathcal{P}}$ and returns a new planset $\widehat{\mathcal{P}}_{new}$ such that $\langle\!\langle \widehat{\mathcal{P}}_{new} \rangle\!\rangle$ is a *subset* of $\langle\!\langle \widehat{\mathcal{P}} \rangle\!\rangle$. $\mathcal{R}$ is said to be a **progressive** refinement if $\langle\!\langle \widehat{\mathcal{P}}_{new} \rangle\!\rangle$ is a proper subset of $\langle\!\langle \widehat{\mathcal{P}} \rangle\!\rangle$. $\mathcal{R}$ is said to be a **complete** refinement if all solutions of $\widehat{\mathcal{P}}$ are also solutions of $\widehat{\mathcal{P}}_{new}$ (i.e., $\langle\!\langle \widehat{\mathcal{P}} \rangle\!\rangle \cap \mathcal{L}(I,G) \equiv \langle\!\langle \widehat{\mathcal{P}}_{new} \rangle\!\rangle \cap \mathcal{L}(I,G)$). $\mathcal{R}$ is said to be **systematic** if given an irredundant planset, it produces another irredundant planset.

Intuitively, a complete and progressive refinement strategy narrows down the candidate set of a planset by winnowing out action sequences that cannot be solutions to the problem. Thus, when we refine the null plan $\mathcal{P}_\emptyset$ repeatedly, we get a sequence of plan sets $\widehat{\mathcal{P}}_i$ which satisfy:[3]

$$\mathcal{U} \equiv \langle\!\langle \mathcal{P}_\emptyset \rangle\!\rangle \supset \langle\!\langle \widehat{\mathcal{P}}_2 \rangle\!\rangle \supset \langle\!\langle \widehat{\mathcal{P}}_2 \rangle\!\rangle \ldots \langle\!\langle \widehat{\mathcal{P}}_n \rangle\!\rangle \ldots \supset \mathcal{L}(I,G)$$

Examples of Refinement Strategies: As shown in [13], existing planners use three types of complete and progressive refinement strategies – forward state space refinement (FSR), backward state space refinement (BSR) and plan space refinement (PSR). Informally, FSR can be understood as eliminating action sequences that have unexecutable prefixes, and BSR can be understood as eliminating action sequences that have infeasible suffixes (in that not all goals can hold at the end of the suffix). Finally, PSR can be understood as eliminating action sequences that do not contain relevant actions. All of them introduce new action constraints onto the plansets and thus increase the length of their minimal candidates. In addition, FSR and BSR add contiguity constraints between actions – thus fixing their relative positions, and giving state information, while PSR adds precedence constraints, fixing only the relative ordering between steps.

In the rocket domain, The FSR refinement takes the singleton planset containing the null plan $\mathcal{P}_\emptyset : \langle 0 \prec \infty \rangle$ and gives the planset:

$$\left\{ \begin{array}{c} \langle 0 * load(A) \prec \infty \rangle, \\ \langle 0 * load(B) \prec \infty \rangle, \\ \langle 0 * Fly() \prec \infty \rangle \end{array} \right\}$$

since only the actions $load(A)$, $load(B)$ and $Fly()$ are applicable in the initial state. It is easy to see that FSR refinement strategy is complete, since every solution to the problem must start with one of the applicable actions. The refinement is progressive since all action sequences belonging to the candidate set of $\mathcal{P}_\emptyset$ which start with unexecutable actions are eliminated after the refinement. Finally, the refinement is systematic since the candidates of the different plansets start with different initial actions. BSR is similar to FSR, except that it extends the suffix of the plan by considering all actions that are applicable in the backward direction.

Finally, PSR refines a planset by "establishing" preconditions of steps in the component plans. Specifically, it picks a precondition C of some step s of a component plan P_i in the given planset $\widehat{\mathcal{P}}$. It then returns a new planset $\widehat{\mathcal{P}}_{new}$ in which P_i is replaced by a set of plans each corresponding to a different way of "establishing" the condition C. In the rocket domain, if we apply PSR refinement to the null plan $\mathcal{P}_\emptyset$, to support the top level goal $At(A,M)$, we generate the singleton planset:

$$\left\{ \left\langle \begin{array}{c} 0 \prec 1 : Fly() \prec \infty, \\ In(A)@1, \\ (1 \overset{At(A,M)}{-} \infty), (1 \overset{\neg At(A,M)}{-} \infty), \end{array} \right\rangle \right\}$$

The point truth constraint $In(A)@1$ is added to force the $Fly()$ action to give the effect $At(A,M)$. The two optional interval preservation constraints are added to ensure that this establishment will not be violated, or repeated, by the actions introduced by later refinements. Once again, it is easy to see that PSR refinement is progressive in that we have eliminated all action sequences that do not contain $Fly()$ action, and complete in that every solution to the problem must satisfy the constraints of the plan shown here.

4 Refinement Planning

The operation of a refinement planner can be understood, broadly, as starting with the null plan $\mathcal{P}_\emptyset$ and repeatedly narrowing down the candidate set by the application of refinement strategies. Figure 3 shows a general refinement planning procedure, Refine($\mathcal{P}$). We will now discuss the development of this procedure.

The simplest case of refinement planning algorithm consists of the procedure in Figure 3, sans the optional steps. This algorithm first checks to see if the current planset is consistent. The second step checks for termination by examining the minimal candidates of the planset to see if any of them correspond to solutions to the problem. Since there are at most exponential number of minimal candidates (corresponding to the safe ground linearizations) for each component of the planset, and since we can check if an action sequence is a solution in linear time, the solution extraction process can be cast as a combinatoric search problem, such as CSP or SAT [9]. The length of the minimal candidates of a plan increase as refinements are applied to it, thus allowing for an incremental exploration of the candidates.

The third step involves refining the planset to generate a new planset. As long as the refinements are complete and progressive, termination is ensured for any solvable problem

[3] Thus, complete and progressive refinements can be seen as computing increasingly finer upper bounds on $L(I,G)$. In [5], Ginsberg presents an novel refinement strategy that simultaneously computes increasingly finer *lower bounds* on $L(I,G)$. In such cases, as soon as the lower bound is non-empty, we can terminate with any of its minimal candidates.

Refine($\widehat{\mathcal{P}}$:Planset)

0. Consistency Check: If $\widehat{\mathcal{P}}$ has no minimal candidates (i.e., safe linearizations), fail.

1. Solution Extraction:
If an action sequence $\langle A_1, A_2, \ldots, A_n \rangle$ is a minimal candidate of $\widehat{\mathcal{P}}$ and also solves the planning problem, terminate and return the action sequence. (Can be cast as a CSP/SAT problem).

2. Refinement: Select refinement R and apply it to $\widehat{\mathcal{P}}$ to get a refined planset $\widehat{\mathcal{P}}_{new}$.

3. Planset splitting (search): (optional) Select a number $k \geq 1$ such that k is less than or equal to the number of components in $\widehat{\mathcal{P}}_{new}$. Split $\widehat{\mathcal{P}}_{new}$ into k plansets $\widehat{\mathcal{P}}_1, \widehat{\mathcal{P}}_2 \ldots \widehat{\mathcal{P}}_k$. *Nondeterministically* select a planset $\widehat{\mathcal{P}}_i$ and set it to $\widehat{\mathcal{P}}_{new}$.

4. Planset simplification (constraint propagation)
(optional) Simplify $\widehat{\mathcal{P}}_{new}$ by enforcing local consistency among constraints. If the simplification shows an inconsistency, then eliminate the plan from further consideration.

5. Recursive invocation: Call $Refine(\widehat{\mathcal{P}}_{new})$.

Figure 3: General Template for Refinement Planning. With k set to the number of components in the planset, we have refinement planning with complete search, as done in most existing planners. With k set to 1, we have refinement planning without search, as is done in GRAPHPLAN, SATPLAN etc. UCPOP-D described in this paper corresponds to a value of k between 1 and the number of planset components.

(in that we will ultimately produce a planset one of whose minimal candidates correspond to solutions).

Introducing search into refinement planning: Most refinement planners augment the pure refinement planning procedure by introducing explicit search into the process. This is done by splitting the components of the plansets into the search space, so that individual components can be refined one by one. In general, splitting the planset components into the search space trades off the search space size increase against the reduction in the solution extraction process. This splitting operation is so prevalent in refinement planners that many previous accounts of refinement planning (including our own [12]) considered the splitting to be a requirement of the refinement planning. As the foregoing discussion shows, this is not necessary.

Controlled search through controlled splitting: Although introducing search in refinement planning reduces the cost of solution extraction function, it does so at the expense of increased search space size. In a way, splitting all the components of a planset into different search branches is an extreme approach for taming the cost of solution extraction. A better solution is to be more deliberate about the splitting.

The algorithm template in Figure 3 supports this by allowing any arbitrary amount of splitting. Specifically, the plan set after refinement, $\widehat{\mathcal{P}}_{new}$, can be split such that some subset of its components are considered together in each search branch.

Handling plansets without splitting: In the foregoing, we have seen that search is introduced into refinement planning by splitting the plansets, and it can be eliminated by handling plansets together. Keeping plansets together and searching for solutions among their minimal candidates also supports a clean separation of planning and scheduling – with refinement strategies doing the bulk of action selection and the CSP/SAT techniques doing the bulk of action sequencing.

There are of course several concerns regarding handling plansets without splitting. The first is that this can lead to very unwieldy data structures. This concern can be alleviated by the use of *disjunctive* plan representations, which allow us to represent a planset containing multiple components by a single partial plan with disjunctive step, ordering, and auxiliary constraints. This representational transformation is best understood as converting external disjunction (among the components of a planset) into internal disjunction (among the constraints of a plan).

The second concern is that avoiding splitting of plansets may just transfer complexity from search space size to plan-handling cost, and thus may not lead to overall improvements in performance in the worst case. We may still hope to win on average for two reasons. First, the CSP and SAT algorithms, which can be used to extract solutions from plansets, seem to scale up much better in practice than general state space search [18, 3], thus encouraging the idea of pushing the complexity into solution extraction phase. Second, and perhaps more important, we can reduce the plan handling costs in disjunctive plan representations by the use of constraint propagation techniques that enforce local consistency among the disjunctive plan constraints. This in turn reduces the number of component plans generated by later refinement strategies.

Thus, although we cannot expect a real performance improvement just by handling plansets without splitting (in that this merely exchanges complexity from search space size to solution extraction cost), disjunctive representations, constraint propagation, and SAT algorithms can in practice tilt the balance in its favor. In the next section, we elaborate the use of disjunctive representations and constraint propagation techniques.

5 Refining Disjunctive Plans and Constraint propagation

The general idea of disjunctive representations is to allow disjunctive step, ordering, and auxiliary constraints into the partial plan representation. Figure 4 shows two examples of

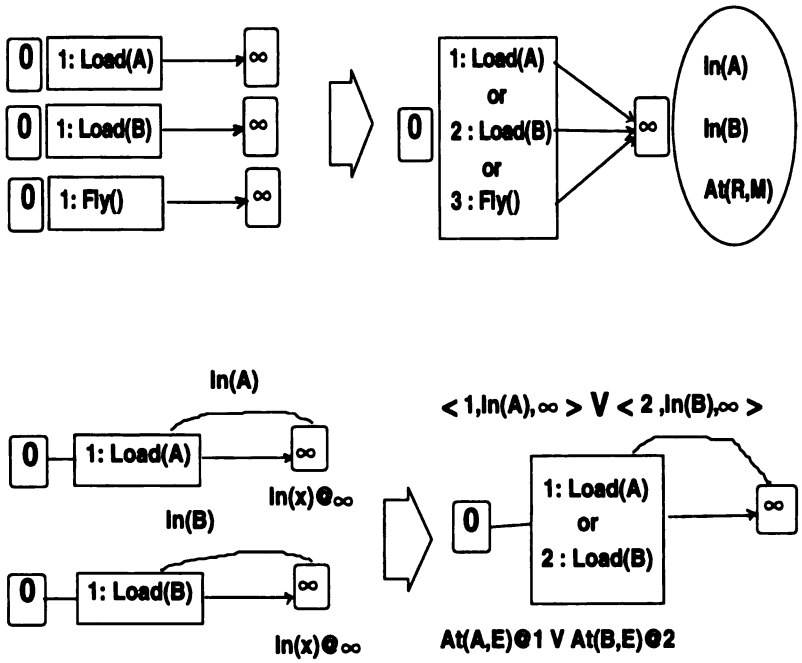


Figure 4: Converting plansets into disjunctive plans. On the top right, there are three plans that may be generated by a forward state space refinement, and on the top left is a single disjunctive plan that is equivalent to the three plans Similarly, on the bottom left we have two partial plans that may be generated by a plan-space refinement. On the top right is a single disjunctive plan that is equivalent to these two plans. .

converting sets of plans into single disjunctive plans. The three plans on the top left can be converted into a single plan on the right, with the disjunctive step and contiguity constraints. Similarly, the two plans on the bottom left can be converted into a single plan on the bottom right with disjunctive step, ordering and auxiliary constraints.

Candidate set semantics for disjunctive plans develop from the simple observation that the set of action sequences that satisfy the disjunctive constraint $c_1 \vee c_2$ is just the union of the set of action sequences that satisfy c_1 and c_2 respectively. In particular, the disjunctive plan on top left in Figure 4 admits into its candidate set any action sequence which starts with $Load(A)$, $Load(B)$ or $Fly()$ actions.

Disjunctive representations clearly lead to a significant increase in the cost of plan handling. For example, in the disjunctive plan on top right side, we do not know which of the steps will be coming next to step 0 and thus we do not quite know what the state of the world will be after the disjunctive step. So, how are we going to apply the FSS refinement? Similarly, in the disjunctive plan on the top right corner in Figure 4, we do not know whether steps 1 or 2 or both will be present in the eventual plan. Thus we do not know whether we should work on $At(A, E)$ precondition or the $AT(B, E)$ precondition.

At first glance, this might look hopeless as the only reasonable way of refining the disjunctive plans will be to split disjunction into the search space again, and refine the components separately. . However, it turns out that we are underestimating the power of what we do know, and how that knowledge can be used to constrain further refinements.

For example, in the plan on top right in Figure 4, knowing that only $Load(A)$, $Load(B)$ or $Fly()$ could have occurred in the first time step lets us realize that any eventual state of the world after the first step will contain only some subset of the conditions $In(A)$, $In(B)$ and $At(R, M)$, plus the conditions true in the initial state. This list of "feasible conditions" is best seen as the "union" of the states after the first time step. It is clear that any action whose preconditions are not a subset of this set cannot be executed in the second time step, *no matter which action we execute in the first time step.* This allows us to do a version of FSR that considers only those actions whose preconditions hold in the current list of feasible propositions at the current time step.

We can do even better in tightening the possible refinements. The knowledge that both $Load(A)$ and $Fly()$ actions cannot occur together in the first time step (since their preconditions and effects interfere), tells us that the second state may either have $In(A)$ or $At(R, M)$ but not both. So, any action whose preconditions include these conditions can also be ignored. This is an instance of propagation of mutual exclusion constraints and can be used to reduce the number of actions considered in the next step by the forward state space refinement. Specifically, the actions that require both $In(A)$ and $AT(R, M)$ can be ignored. This type of constraint prop-

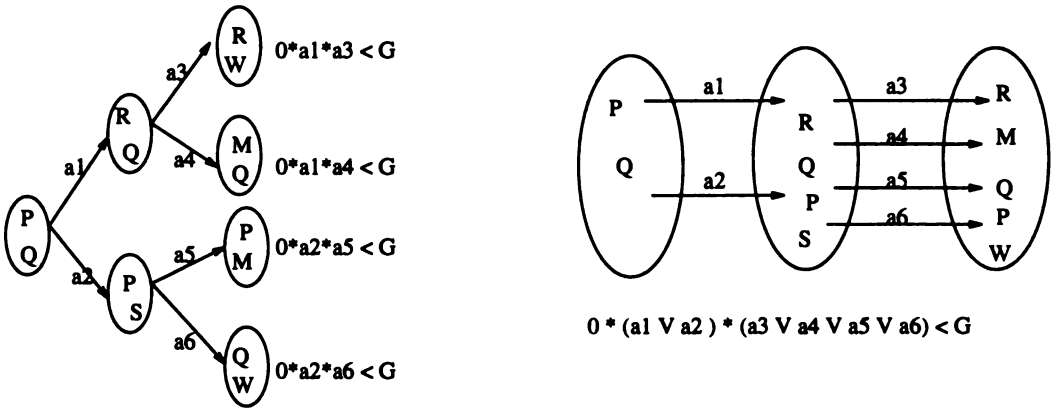


Figure 5: Interpreting GRAPHPLAN in terms of disjunctive representations. To the left is the search space generated by a forward state space refinement. To the right is the partial plan representation, called plan-graph, used in GRAPHPLAN Each candidate plan of the plan graph must have some subset of the actions in i^{th} level coming immediately before some subset of actions in the $i+1^{th}$ level (for all i). The minimal candidates corresponding to all plans generated by the forward state space planner are compactly represented by a single partial plan (plan graph) in GRAPHPLAN.

agation forms the backbone of the GRAPHPLAN algorithm [2] planning algorithm, that we shall discuss in more detail below. Of course, unless the propagation is very strong, we will not be able to weed out all infeasible actions from being considered. In summary, we can do refinements on the disjunctive representations directly, if we do not mind admitting some action sequences that would not have been the candidates of a planset produced by the same refinement operating on non-disjunctive plans. Soundness can still be maintained since the solution extraction process checks to make sure that a minimal candidate is a solution, before terminating.

Similar ideas apply to the disjunctive plan on the bottom right in Figure 4. For example, knowing that either 1 or 2 must precede the last step and give the condition tells us that if 1 does not, then 2 must. This is an instance of constraint propagation on orderings and reduces the number of establishment possibilities that plan-space refinement has to consider at the next iteration. We will see an example of this in UCPOP-D system described in Section 6. It is worth noting that the advantages of constraint propagation depend critically on the disjunctive representations. If we represented the plansets in terms of their components, the constraints on the different plan components would have been in terms of different step names and would thus not have interacted.

5.1 Case Study: GRAPHPLAN

GRAPHPLAN [2] is a recent planning system, that can be understood as using a partial plan representation that corresponds to the disjunction of the refinements produced by a forward state space planner (see Figure 5)[4]. Specifically, GRAPHPLAN's planning process involves two phases that are alternated until a plan is found. In the first phase, a compact structure called "plan-graph" is constructed. A plan-graph corresponds to the disjunction of all the refinements produced by a forward state space planner [16]. Thus the GRAPHPLAN refinement process does not introduce any branching into the search space. All the complexity is transferred to the solution extraction process which has to search the plan graph structure for minimal candidates that are solutions. A plan-graph specifies sets of operators at different time steps, such that each candidate solution must contain a subset of the actions at each time step contiguous to each other. As illustrated in Figure 5, the plan-graph can be seen as a disjunctive representation of the plansets generated by forward state space refinements. Empirical results show that GRAPHPLAN scales-up significantly better than non-disjunctive planners on a large number of benchmark domains.

GRAPHPLAN depends crucially on constraint propagation routines both in the plan-graph construction and solution extraction phases. Constraint propagation is done in terms of mutual exclusion relations between actions, stating that the presence of one action might necessitate the absence of another action, or vice versa. Two actions are mutually exclusive if their preconditions and effects are mutually conflicting. The mutual exclusion of actions at one time step can be propagated to make otherwise independent actions at a latter time step mutually exclusive. Extension of k-level plan-graph to a $(k+1)$-level plan-graph roughly corresponds to doing forward state-space refinement on all the components of the k^{th} level planset.[5] Empirical results demon-

[4]There are some minor further differences between the search space of normal forward state space refinements and the plan-graph representation of GRAPHPLAN. Specifically, plan-graph construction is better understood in terms of a forward state space planner which allows "noop" actions, and projects sets of independent operators simultaneously from the current state See [16] for a full reconstruction of GRAPHPLAN from forward projection.

[5]Strictly speaking, the candidate set of the k-level plangraph is a *superset* of the candidate set of the corresponding k^{th} level planset [16]. This is because GRAPHPLAN may introduce actions into level $k+1$ which may not be applicable in any physical state af-

strate that shifting the complexity entirely from search space size to solution extraction this way does lead to significant improvements in performance. Kautz and Selman [9] show that GRAPHPLAN's performance can be further improved by casting its solution extraction process as a SAT problem and solving it using local search methods.

6 UCPOP-D: Disjunction over Plan-space refinements

In the foregoing, we have argued in favor of disjunctive refinements and constraint propagation, and suggested that GRAPHPLAN algorithm can be understood in these terms. The success of GRAPHPLAN shows that there is a lot to be gained by considering other disjunctive partial plan representations. Since GRAPHPLAN concentrates on disjunctive representations of a forward state space planner, one interesting direction is to explore disjunctive representations over other refinements. In this section, we will describe our experience with disjunctive representations over plan-space refinements in the context of UCPOP [19], a popular partial-order planner.

This section has multiple aims. We want to demonstrate that the ideas of disjunctive representations can be folded naturally into existing refinement planners. We also want to explore the middle-ground in terms of splitting plansets. UCPOP splits every component of the planset resulting from a refinement into the search space. GRAPHPLAN keeps all components together with disjunctive representations. The UCPOP-D algorithm we discuss here disjoins some of the components of the planset produced by a refinement into a single plan, while keeping the other components separately.

Since plan-space refinements concern themselves with different ways of establishing a specific condition in the plan, disjunction here will deal with multiple establishment possibilities together. Let us motivate the utility of such a disjunction with an example. Consider the plan shown to the left in Figure 6, which contains a step s that requires a precondition p. There are two steps s_1 and s_2 in the plan such that both of them are capable of providing the condition p to s. When planners using plan-space refinement consider the precondition $p@s$ for establishment, they typically make several refinements, two of them corresponding to simple establishment with s_1 and s_2 respectively. The resulting refinements will contain IPCs $(s_1 \overset{p}{-} s)$ and $(s_2 \overset{p}{-} s)$ respectively. One way of cutting down the branching in this process is to combine these two plans into a single refinement, and ensure that either s_1 or s_2 will give p to s.[6] The search space schematic on the right of Figure 6 illustrates how the branching factor is reduced by such disjunctive causal commitment constraints. Specifically, the simple establishment options are all bundled into a single plan.

To support such disjunctive causal commitments, we have to ensure that (a) either s_1 or s_2 will precede s and (b) for every step s_t that deletes p, either either s_t comes after s or it comes before either s_1 or s_2, with s_1 or s_2 preceding s at the same time. More generally, if we want to use any of n steps $s_1, s_2 \cdots s_n$ to support some condition at step s, we need to impose the following disjunctive ordering constraints:

$$(s_1 \prec s) \vee \ldots \vee (s_n \prec s).$$

For every step s_t that can threaten the establishment, we need:

$$(s \prec s_t) \vee [(s_t \prec s_1) \wedge (s_1 \prec s)] \vee \ldots \vee [(s_t \prec s_n) \wedge (s_n \prec s)].$$

6.1 Handling Disjunctive Orderings via constraint propagation

In the above, we noticed that maintaining disjunctive causal structures ultimately boils down to handling disjunctive orderings (in the case of propositional planning). This can be done efficiently with the help of constraint propagation techniques [20]. The basic idea is that whenever an atomic ordering constraint is added to the plan, it can be propagated through the disjunctive constraints, simplifying them. The simplification may give rise to more atomic orderings, which in turn cause further simplifications. In our implementation, we do two types of direct simplifications or local consistency enforcement: (1) A disjunctive ordering constraint $O_1 \vee O_2 \ldots \vee O_n$ (where O_i are atomic orderin constraints) simplifies to $O_1 \vee O_2 \vee \ldots O_{i-1} \vee O_{i+1} \ldots \vee O_n$ if an ordering O_i^{-1} (where the inverse of an ordering relation $s_1 \prec s_2$ is $s_2 \prec s_1$) is propagated through it. It also simplifies to $True$ when an ordering O_i is propagated through it.[7] This type of propagation often simplifies the disjunctions completely. If disjunctive orderings remain in the plan by the time all other open conditions are established, we can solve the CSP corresponding to the ordering constraints to check if there exists a ground linearization of the plan that is consistent. Rather than use a separate CSP, in our current implementation, we simply split the disjunction into the UCPOP search space. Since we only do this splitting for those disjunctive orderings that remain unsimplified, the search space size is likely to be much smaller than that for normal UCPOP.

---

ter the components of the k^{th} level planset. Some, but not all, of these inapplicable actions are weeded out by the fact that GRAPHPLAN propagates mutual exclusion relations among state literals and avoids introducing actions whose preconditions are mutually exclusive. This is the price we pay for the simplicity of disjunctive representation.

[6] The idea of maintaining multiple causal contributors has been around, and our previous work provides a formalization of them [11], and uses them to revoke prior causal commitments. However, this is the first time that disjunctive causal commitments are used in their full generality, involving disjunctive ordering constraints and constraint propagation, to control search space explosion

[7] It is of course possible to enforce stronger constraint propagation – for example, resolving two disjunctive ordering constraints $O_1 \vee O_2$ and $O_1' \vee O_2'$ to $O_1 \vee O_2'$ when $O_2 = \neg O_1'$. But, our current experience is that such stronger propagation strategies do not improve performance [20].

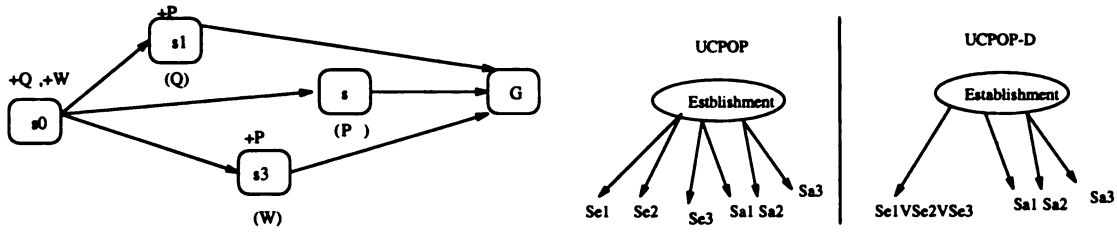


Figure 6: Combining establishment branches in plan-space refinement using disjunction

6.2 Implementation and Empirical Evaluation

We implemented these ideas about disjunctive causal constraints on top of the standard UCPOP system from University of Washington [19]. Our initial implementation made minimal changes to UCPOP – for example, we depend on the standard termination criterion of UCPOP, rather than the minimal candidate based termination. For convenience, we will call this variant of UCPOP, UCPOP-D. Since UCPOP is already optimized to handle consistency with atomic ordering constraints, the easiest way to make it handle disjunctive orderings was to keep a separate field in the plan structure for disjunctive orderings. Whenever propagation techniques derive new atomic orderings from the existing disjunctive orderings, they are added to the normal orderings list of UCPOP. As described above, the changes to the algorithm come in two places – first is in establishing new conditions through existing steps, and the second is in handling of conflicts to causal commitments. In both cases, additional disjunctive orderings are added to the plan representation.

The disjunctive orderings are simplified whenever the planner adds non-disjunctive orderings to the plan. The latter are added by the planner either as part of handling unsafe links when only one contributor or one type of conflict resolution is possible. In either case, the ordering is *propagated* through the disjunctive orderings of the plan.

In most cases, propagation simplifies disjunctive orderings of the plan so that by the time all open conditions and unsafe links are handled, the plan will not have any disjunctive orderings left. If disjunctive orderings are left unsimplified however, the normal termination criteria of UCPOP will not apply and unsound plans can be returned. In the current implementation, we avoid this by explicitly splitting the disjunction into the search tree in such cases.

6.3 Results

To see whether or not the use of disjunctive constraints and propagation helps improve the planner efficiency, we conducted empirical studies in a variety of domains. To provide a baseline for comparison, we used three different versions of UCPOP - one with simple causal constraints, but with constraint propagation used to handle conflict resolution (CP+NMCL), the second with disjunctive causal commitments and constraint propagation over disjunctive orderings (CP+MCL) and finally the standard UCPOP implementation, which uses single contributor causal links and non-disjunctive representations. The plots in Figure 7 show the results of these experiments. The first set of plots show the comparative performance in ART-MD-RD domain, which was used in [11] to illustrate the utility of multi-contributor causal links. The second set of plots show the results in Link-Chain domain, which was used by Veloso et. al [21] to highlight the inadequacies of the single-contributor causal structures. Finally, the third set of plots show the results in the prodigy blocks world domain (with *pickup*, *putdown*, *stack* and *unstack* actions). The plots show that in all cases, disjunctive representations and constraint propagation outperform standard UCPOP. In particular, the disjunctive causal commitments improve performance significantly.

6.4 Extending UCPOP-D

UCPOP-D can be extended in several ways. The first obvious step would be to extend it to non-propositional cases, and this can be done by applying constraint propagation to variable codesignation and non-codesignation constraints. Allowing disjunction over step-addition establishments, and/or handling actions with conditional effects will lead us to disjunctive open conditions and will necessitate more extensive changes to partial-order planning algorithms. The main changes come in the form of generalizing plan-space refinement to handle fully disjunctive partial plans. For example, suppose we are considering two contributors s_1 and s_2 as possible contributors of the condition p to the step s. Suppose s_1 has a conditional effect $r \Rightarrow p$ and s_2 has the conditional effect $q \Rightarrow p$. Since $r@s_1$ will be a secondary precondition if s_1 gives p and $q@s_2$ will be a precondition if s_2 gives p, the disjunctive causal commitment thus leads to disjunctive open condition $r@s_1 \vee q@s_2$, only one of which need to be achieved. Even here, it is possible to handle them in terms of steps currently existing in the plan. Suppose the steps s_3 and s_4 are currently in the plan and they give conditions r and q respectively, we can take care of the disjunctive open condition $r@s_1 \vee q@s_2$, with the help of the disjunctive causal constraint $(s_3 \overset{r}{-} s_2) \vee (s_4 \overset{q}{-} s_3)$, and handling it in the usual way. The problem comes when we are trying to make the disjunctive open condition true by adding new steps. From least commitment point of view, it is best to introduce all the new steps capable of establishing any of the

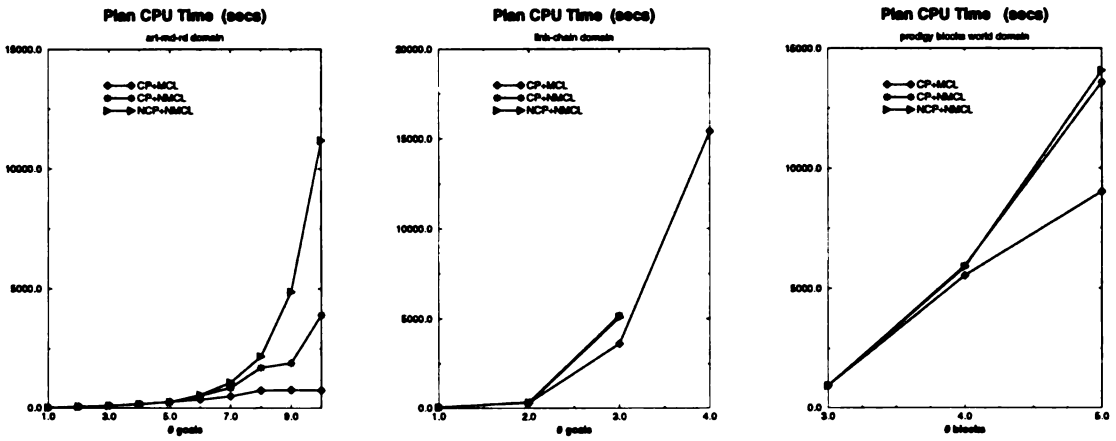


Figure 7: Plots showing the utility of disjunctive causal link representations in multiple domains

disjuncts of the disjunctive open condition simultaneously, and consider the steps to be "tentative" in that only one of those steps need be present in the final plan. The tentative steps will then give rise to tentative open conditions which need to be handled. While applying establishment refinement to all tentative conditions is one possibility – and this is essentially what is done in the causal "SNLP" encodings used in [10] – we are hopeful that there may be ways of more tightly constraining the refinements (as is done in GRAPHPLAN using mutual exclusion constraints, see Section 5).

7 Discussion

In this section, we shall discuss several broad issues raised by planning with disjunctive representations:

Amount of splitting: Research in the constraint satisfaction literature shows that propagation and splitting can have synergistic interactions in improving planner performance. Specifically, splitting leads to commitment, which in turns supports local consistency enforcement. As an example, in the 8-queens problem, committing the position of one of the queens leads to improved local consistency enforcement through arc-consistency algorithms. This leads to the possibility that the best way to do planning may involve being deliberate about splitting the plansets into the search space. Complete splitting leads to search space explosions as typified in UCPOP type planners, while avoiding splitting all together may inhibit constraint propagation.

Table 1 describes a variety of planners and the amount of splitting/search they employ. The question of how much splitting is to be done is a hard one. We believe that to a large extent this depends on common substructure between the components of the planset. In general, it may be best to combine components with shared structure into a single disjunctive plan so that propagation is facilitated among its constraints. UCPOP-D provides a preliminary example of this strategy.

Relative support provided by different refinements: Another important open issue is which refinements should we be using to get the maximum mileage out of disjunctive representations. Clearly, all refinements can support planning with disjunctive representations. We have already seen that GRAPHPLAN can be understood in terms of disjunction over state space refinements, while UCPOP-D and Descartes [8] can be understood in terms of disjunction over plan-space refinements. An important issue is the relative tradeoffs offered by disjunction over different types of refinements. The previous analyses of relative tradeoffs concentrated on the level of commitment enforced by a refinement strategy in its planset components and on the subgoal interactions that arise due to this. This analyses is of little use in planners using disjunctive representations since they do not uniformly split planset components into the search space. Instead, we need to concentrate on issues such as: (a) the efficiency of extraction of solutions from the plansets and (b) the support for constraint propagation provided by the plansets. As an example of such a tradeoff, recent work by Kautz et. al. [10] indicates that SAT "encodings" based on disjunctive state-space refinements are larger than the encodings based on plan-space refinements, but that the latter do not seem to be as easy to handle during solution extraction process.

8 Related Work

As we mentioned, our description of refinement planning is general enough to include both traditional refinement planners and some new planning algorithms. Table 1 characterizes several of these planners in terms of our framework. We believe that the understanding of the role of disjunctive refinements and constraint propagation that we developed in this paper facilitate a clearer appreciation of the connections between the many CSP-based planning algorithms and traditional refinement planning. We have already shown how GRAPHPLAN can be understood in terms of re-

Planner	Level of splitting	Type of refinement
UCPOP [19], SNLP [17]	Full	PSR
TOPI [1]	Full	BSR
GRAPHPLAN [2]	No	FSR
SATPLAN [9]	No	FSR/PSR
Descartes [8]	Some	PSR
UCPOP-D	Some	PSR

Table 1: Different planners as instantiations of **Refine** algorithm template

finement planning with disjunctive representations and constraint propagation. Another successful strand of work in plan generation is related to SATPLAN, which considers planning as a satisfiability problem [9, 10]. SATPLAN starts with a SAT encoding such that all k-step solutions to the planning problem correspond to the satisfying assignments of the encoding. The idea is to produce SAT encodings corresponding to different solution lengths, and solve them until a satisfying assignment is found to one of the instances. Much of the work here is aimed at generating "compact" encodings that are easy to solve using existing systematic or local satisfiability checking algorithms.

Described this way, there seems to be little connection between this strand of work and the traditional refinement planning algorithms. However, general refinement planning framework presented in this paper does clarify some of the tradeoffs in posing planning as satisfiability. For example, there is a straightforward connection between the SAT encodings for k-step solutions, and the plansets produced by refinement planners. In particular, consider a planset $\widehat{\mathcal{P}}$ produced by any refinement planner using a series of complete and progressive refinements, such that all components of $\widehat{\mathcal{P}}$ have exactly k steps. It can be shown that each k-step solution for the planning problem must correspond to one of the minimal candidates of $\widehat{\mathcal{P}}$.[8] The problem of finding a minimal candidate of $\widehat{\mathcal{P}}$ that corresponds to a solution can be posed as a SAT problem.

This relationship brings up several points to the fore: The exact representation of the planset $\widehat{\mathcal{P}}$, and consequently the size of its SAT encoding, depend on the types of refinements used (recall that we can use any sequence of state space or plan space refinements since they are all progressive and complete). This relates the syntactic notion of the SAT encoding size to the well-established idea of refinement strategies. Different sequences of refinements will correspond to different encodings. There may thus be a strong connection between the theories of refinement selection and the theories of encoding generation.

The current implementation of SATPLAN does not explicitly consider refinements, but rather attempts to come up with a necessary and sufficient set of propositional constraints that must be satisfied by all k-step solutions. It would be interesting to consider generating the encodings using refinements on disjunctive plans. Kautz and Selman [9] do this implicitly when they convert GRAPHPLAN's plan-graph into a SAT instance. This raises the possibility that the research on refinement selection, such as "goal order heuristics" in partial order planning [7], or planning by interleaving multiple refinements [13] may have an impact on generating efficient SAT encodings.

It is also worth understanding the relation between least commitment, task reduction ideas in traditional refinement planning, and the idea of planning with disjunctive representations. Most existing refinement planners use plan representations that comprise of conjunctions of atomic constraints: steps, orderings between steps etc. The usual way of increasing the candidate set size of partial plans (and thus reduced commitment), is to use weaker atomic constraints (e.g., partial ordering instead of contiguity ordering relations between steps). Disjunctive representations provide an alternate way of achieving least commitment and search space reduction. HTN planning [14] can be seen as an idea closely related to disjunctive refinements. In particular, HTN planners introduce non-primitive actions into the partial plans, and gradually reduce them with the help of user-supplied task reduction schemas into primitive actions. Thus, the presence of non-primitive action in a partial plan can be interpreted as the presence of disjunction of all the plan fragments that the action can be eventually reduced to. One important difference is that HTN planners never explicitly deal with this disjunction, but push it gradually into the search space (by considering each reduction of the action in a different search branch). Thus, they miss out on the advantages of handling plansets together, such as constraint propagation. In a way, both traditional least commitment and task reduction ideas can be seen as instances of the general heuristic of converting the search space to have lower branching at the top levels. Disjunctive representations give this ability, but they also support handling plansets together without ever pushing them back into the search space.

9 Conclusion

By teasing apart the hitherto closely intertwined notions of "refinement" and "search", we were able present a model of refinement planning that not only includes the traditional planners, but also supports a large variety of planners that transfer search to solution extraction phase by handling sets

[8] Not all minimal candidates may be k-step solutions, however.

of partial plans together. We have argued that efficient handling of large plansets requires the use of disjunctive plan representations and constraint propagation techniques.

We have used our model of refinement planning to explain the operation of several newer planning approaches such as GRAPHPLAN and SATPLAN. We have also shown how the ideas of disjunctive representations and constraint propagation can be incorporated into traditional planners by presenting the UCPOP-D algorithm that supports disjunctive handling of plans differing only in their causal structures, and demonstrating that UCPOP-D outperforms UCPOP in several domains. Finally, we discussed several broad issues raised by this "inclusive" picture of refinement planning.

Acknowledgements:

We thank Biplav Srivastava for his many helpful critiques of the ideas in this paper, and his help in preparing some figures. This research is supported in part by NSF young investigator award (NYI) IRI-9457634, ARPA/Rome Laboratory planning initiative grants F30602-93-C-0039 and F30602-95-C-0247, and the ARPA ASERT award DAAH-04-96-1-0247.

References

[1] A. Barrett and D. Weld. Partial Order Planning: Evaluating Possible Efficiency Gains. *Artificial Intelligence*, Vol. 67, No. 1, 1994.

[2] A. Blum and M. Furst. Fast planning throug planning graph analysis. In *Proc. IJCAI-95*, 1995.

[3] J.D. Crawford and L.D. Auton. Experimental results on the crossover point in satisfiability problems. In *Proc. AAAI-93*, 1993.

[4] E. Fink and M. Veloso. Formalizing the Prodigy Planning Algorithm. CMU CS Tech. Report, Fall 1994.

[5] M. Ginsberg. A new algorithm for generative planning. In *Proc. KRR-96*, 1996

[6] L. Ihrig and S. Kambhampati. Design and Implementation of a Replay Framework based on a Partial order Planner. In *Proc. AAAI-96*, 1996.

[7] D. Joslin and M. Pollack. Least-cost flaw repair: A plan refinement strategy for partial order planning. *Proceedings of AAAI-94*, 1994.

[8] D. Joslin and M. Pollack. Passive and active decision postponement in plan generation. In *Proc. 3rd European Workshop on Planning*, 1995.

[9] H. Kautz and B. Selman. Pushing the Envelope: Planning, Propositional Logic, and Stochastic Search In *Proc. AAAI-96*, 1996.

[10] H. Kautz, D. McAllester and B. Selman. Encoding plans in propositional logic In *Proc. KRR-96*, 1996.

[11] S. Kambhampati. Multi-Contributor Causal Structures for Planning: A Formalization and Evaluation. *Artificial Intelligence*, Vol. 69, 1994. pp. 235-278.

[12] S. Kambhampati, C. Knoblock and Q. Yang. Planning as Refinement Search: A Unified framework for evaluating design tradeoffs in partial order planning. *Artificial Intelligence* special issue on Planning and Scheduling. Vol. 76. 1995.

[13] S. Kambhampati and B. Srivastava. Universal Classical Planner: An algorithm for unifying state space and plan space approaches. In *Proc European Planning Workshop*, 1995.

[14] S. Kambhampati and B. Srivastava. Unifying classical planning approaches. ASU-CSE-TR 96-006. Submitted for journal publication.

[15] S. Kambhampati, S. Katukam, Y. Qu. Failure Driven Dynamic Search Control for Partial Order Planners: An explanation-based approach *Artificial Intelligence* (To appear in Fall 1996)

[16] S. Kambhampati. Reconstructing GRAPHPLAN Algorithm from Forward Projection. In Planning Methods in AI (Notes from ASU Planning Seminar). ASU CSE TR 96-004. *http://rakaposhi.eas.asu.edu:8001/yochan.html*

[17] D. McAllester and D. Rosenblitt. Systematic Nonlinear Planning. In *Proc. 9th AAAI*, 1991.

[18] B. Selman, H.J. Levesque and D.G. Mitchell. GSAT: A new method for solving hard satisfiability problems. In *Proc. AAAI-92*, 1992.

[19] J.S. Penberthy and D. Weld. UCPOP: A Sound, Complete, Partial Order Planner for ADL. In *Proc. KR-92*, 1992.

[20] E. Tsang. *Foundations of Constraint Satisfaction.* (Academic Press, San Diego, California, 1993).

[21] M. Veloso and J. Blythe. Linkability: Examining causal link commitments in partial-order planning. *Proceedings of AIPS-94*, 1994.

[22] Q. Yang. A theory of conflict resolution in planning. *Artificial Intelligence*, 58:361-392, 1992.

Constraints

Symmetry-Breaking Predicates for Search Problems

James Crawford
Matthew Ginsberg
Computational Intelligence Research Laboratory
1269 University of Oregon
Eugene, OR 97403-1269
{jc, ginsberg}@cirl.uoregon.edu

Eugene Luks
Amitabha Roy
Department of Computer Science
The University of Oregon
Eugene, OR 97403-1202
{luks, aroy}@cs.uoregon.edu

Abstract

Many reasoning and optimization problems exhibit symmetries. Previous work has shown how special purpose algorithms can make use of these symmetries to simplify reasoning. We present a general scheme whereby symmetries are exploited by adding "symmetry-breaking" predicates to the theory. Our approach can be used on any propositional satisfiability problem, and can be used as a pre-processor to any (systematic or non-systematic) reasoning method. In the general case adding symmetry-breaking axioms appears to be intractable. We discuss methods for generating partial symmetry-breaking predicates, and show that in several specific cases symmetries can be broken either fully are partially using a polynomial number of predicates. These ideas have been implemented and we include experimental results on two classes of constraint-satisfaction problems.

1 Introduction

Human artifacts from chess boards to aircraft exhibit symmetries. From the highly regular patterns of circuitry on a microchip, to the interchangeable pistons in a car engine, or seats in a commercial aircraft, we are drawn esthetically and organizationally to symmetric designs. Part of the appeal of a regular or symmetric design is that it allows us to reason about and understand larger and more complex structures than we could otherwise handle. It follows that if we are to build computer systems that configure, schedule, diagnose, or otherwise reason about human artifacts, we need to endow these reasoning systems with the ability to exploit structure in general and symmetries in particular.

Automated reasoning is a huge area, so for purposes of this paper we will focus on search. Abstractly, a search problem consists of a large (usually exponentially large) collection of possibilities, the *search space*, and a predicate. The task of the search algorithm is to find a point in the search space that satisfies the predicate. Search problems arise naturally in many areas of Artificial Intelligence, Operations Research, and Mathematics.

The use of symmetries in search problems is conceptually simple. If several points in the search-space are related by a symmetry then we never want to visit more than one of them. In order to accomplish this we must solve two problems. First, the symmetries need to be discovered; e.g., we need to realize that we can interchange the five ships without changing the basic form of the problem. Second, we need to make use of the symmetries.

This paper focuses on the second of these problems. It was shown by Crawford [Crawford, 1992] that detecting symmetries is equivalent to the problem of testing graph isomorphism, a problem that has received a substantial amount of study (see, *e.g.*, [Babai, 1995]).

With regard to taking computational advantage of the symmetries, past work has focused on specialized search algorithms that are guaranteed to examine only a single member of each symmetry class [Brown *et al.*, 1988; Crawford, 1992]. Unfortunately, this makes it difficult to combine symmetry exploitation with other work in satisfiability or constraint satisfaction, such as flexible backtracking schemes [Gaschnig, 1979; Ginsberg, 1993] or nonsystematic approaches [Minton *et al.*, 1990; Selman *et al.*, 1992]. Given the rapid progress in search techniques generally over the past few years, tying symmetry exploitation to a specific search algorithm seems premature.

The approach we take here is different. Rather than modifying the search algorithm to use symmetries, we will use symmetries to modify (and hopefully simplify) the problem being solved. In tic-tac-toe, for example, we can require that the first move be in the middle, the upper left hand corner, or the upper middle (since doing this will not change our analysis of the game in any interesting way). In general, our approach will be to add additional constraints, *symmetry-breaking predicates*, that are satisfied by exactly one member of each set of symmetric points in the search space. Since these constraints will be in the same language as the original problem (propositional satisfiability for purposes of this paper) we can run the symmetry detection and utilization algorithm as a preprocessor to any satisfiability checking algorithm.

Of course there is a catch. In this case two catches: First, there is no known polynomial algorithm for detecting the symmetries. Symmetry detection is equivalent to graph isomorphism which is believed to be easier than NP-complete, but is not known to be polynomial. Nevertheless, graph isomorphism is rarely difficult in practice, as has been profoundly demonstrated by the efficient nauty system [McKay, 1990]. Furthermore, it has been shown that, on average, graph isomorphism is in linear time using even naive methods [Babai and Kučera, 1979]. The second catch is that even after detection is complete, computing the full symmetry-breaking predicate appears to be intractable.

However, there is generally no reason to generate the full symmetry breaking predicate. We can generate a partial symmetry-breaking predicate without affecting the soundness or completeness of the subsequent search. We will show that in several interesting cases we can break symmetries either fully or partially using a polynomial number of predicates.

The outline of the rest of this paper is as follows: we first define symmetries of search problems, and discuss how predicates can be added to break symmetries. We then discuss both exact and partial methods for controlling the size of the symmetry-breaking predicate. Finally we discuss experimental results and related work.

2 Definitions and Preliminaries

For purposes of this paper we will assume that we are working in clausal propositional logic. The symmetries of a propositional theory will be defined to be the permutations of the variables in the theory that leave the theory unchanged. These symmetries form a group and we use techniques and notation from computational group theory throughout the paper.

Let L be a set of propositional variables. As usual, literals are variables in L, or negations of variables in L. If $x \in L$, then we write the negation of x as $\overline{x}$. A clause is then just a disjunction of literals (written, *e.g.*, $x \vee y \vee z$), and a theory is a conjunction of clauses. One basic observation that will be critical to the definitions below is that two clauses are considered to be identical iff they involve the same set of literals (*i.e.*, order is not significant) and two theories are identical iff they involve the same set of clauses.

A *truth assignment* for a set of variables L is a function $A : L \to \{t, f\}$ (on occasion we write $1, 0$ for t, f respectively). In the usual way, A extends by the semantics of propositional logic to a function on the set of theories over L, and, by abuse of notation, we continue to denote the extended function by A.[1] A truth assignment A of L is called a *model* of the theory T if $A(T) = t$. The set of models of T is denoted $\mathcal{M}(T)$.

The propositional satisfiability problem is then just (see, *e.g.*, [Garey and Johnson, 1979]):

Instance: A theory T.

Question: Is $\mathcal{M}(T)$ non-empty (*i.e.*, does T have a model)?

Clearly one can determine whether such an assignment exists by trying all possible assignments. Unfortunately, if the set L is of size n then there are 2^n such assignments. All known approaches to determining propositional satisfiability are computationally equivalent (in the asymptotically worst case) to such a complete search. Propositional satisfiability is thus one of the simplest "canonical" examples of a search problem.

To formally define symmetries we need some additional notation. Consider a set L. The group of all permutations of L is denoted by $\mathrm{Sym}(L)$.[2] This is a group under composition; the product $\theta\phi$ of $\theta, \phi \in \mathrm{Sym}(L)$ is taken to be the result of performing θ and then ϕ. If $v \in L$ and $\theta \in \mathrm{Sym}(L)$, the image of v under θ is denoted v^θ (it is standard to write the permutation as a superscript so that we can make use of the natural equality $v^{\theta\phi} = (v^\theta)^\phi$). A permutation θ

[1] That is, $A(\overline{x})$ is the negation of $A(x)$, A is true of a clause iff it is true of at least one of the terms in the clause, and A is true of a theory iff it is true of all the clauses in the theory.

[2] Recall that a permutation of a finite set L is a one-to-one mapping $\theta : L \to L$.

of a set L of variables naturally extends to a permutation of negated variables such that $\overline{v}^\theta = \overline{v^\theta}$, and thus to a permutation of the set of clauses over L, wherein if $C = \bigvee_{i=1}^{r} v_i$, then $C^\theta = \bigvee_{i=1}^{r} v_i^\theta$, and finally to a permutation of the theories over L, namely, if $T = \{C_i\}_{1 \le i \le m}$, then $T^\theta = \{C_i^\theta\}_{1 \le i \le m}$.

Let T be a theory over L and let $\theta \in \mathrm{Sym}(L)$. We say that θ is a *symmetry*, or automorphism, of L iff $T^\theta = T$. The set of symmetries of T is a subgroup of $\mathrm{Sym}(L)$ and is denoted by $\mathrm{Aut}(T)$.

For example, consider the following theory: $a \vee \overline{c}$, $b \vee \overline{c}$, $a \vee b \vee c$, $\overline{a} \vee \overline{b}$. Notice that if we interchange a and b the theory is unchanged (again, only the order of the clauses and the order of literals within the clauses is affected). It is customary to denote this particular symmetry by $(a\ b)$.

Permutations of variables in general, and symmetries in particular, can be viewed as acting on assignments as well as theories. If $\theta \in \mathrm{Sym}(L)$ then θ acts on the set of truth assignments by mapping $A \mapsto {}^\theta A$, where ${}^\theta A(v) = A(v^\theta)$ for $v \in L$.[3] Hence, if T is a theory over L, $A(T^\theta) = {}^\theta A(T)$. Thus, we have the immediate consequence that any symmetry of T maps models of T to models of T, and non-models of T to non-models:

Proposition 2.1 *Let T be a theory over L, $\theta \in \mathrm{Aut}(T)$, and A a truth assignment of L. Then $A \in \mathcal{M}(T)$ iff ${}^\theta A \in \mathcal{M}(T)$.*

More generally, $\mathrm{Aut}(T)$ induces an equivalence relation on the set of truth assignments of L, wherein A is equivalent to B if $B = {}^\theta A$ for some $\theta \in \mathrm{Aut}(T)$; thus, the equivalence classes are precisely the *orbits* of $\mathrm{Aut}(T)$ in the set of assignments. Note, further, that any equivalence class either contains only models of T, or contains no models of T. This indicates why symmetries can be used to reduce search: we can determine whether T has a model by visiting each equivalence class rather than visiting each truth assignment.

3 Symmetry-Breaking Predicates

The symmetry-breaking predicates are chosen such that they are true of exactly one element in each of the equivalence classes of assignments generated by the symmetry equivalence. For example, for the small example theory discussed in section 2, the two models are (t, f, f) and (f, t, f). The theory has one nontrivial symmetry – the interchange of a and b. As required by proposition 2.1, applying this perturbation to a model yields a model. We can "break" the symmetry by adding the axiom $a \rightarrow b$ which eliminates one of the models, (t, f, f), leaving us with only one model from the equivalence class.

In general, we introduce an ordering on the set of variables, and use it to construct a lexicographic order on the set of assignments. We will then add predicates that are true of only the smallest model, under this ordering, within each equivalence class.[4] Intuitively we do this by viewing each model as a binary number (*e.g.*, (t, f, f) would be seen as 100). We then add predicates saying that θ does not map M to a smaller model (for all symmetries θ).

If $\theta \in \mathrm{Sym}(L)$, then θ acts on any sequence of variables in L: if $V = (v_1, \ldots, v_m)$ then $V^\theta = (v_1^\theta, \ldots, v_m^\theta)$. For a sequence $V = (v_1, \ldots, v_m)$ and $0 \le i \le m$, it is convenient to denote by V_i the initial segment $(v_1, \ldots, v_i)$ (of course, this is the empty sequence () if $i = 0$).

If $v, w \in L$, we write $v \le w$ as a shorthand for the clause $v \rightarrow w$. If $V = (v_1, \ldots, v_m)$ and $W = (w_1, \ldots, w_m)$ are sequences of variables in L and $0 \le i \le m$, we let $P_i(V, W)$ abbreviate the predicate

$$V_{i-1} = W_{i-1} \rightarrow v_i \le w_i.$$

Finally, we write $V \le W$ as shorthand for

$$\bigwedge_{i=1}^{m} P_i(V, W),$$

that is,

$$\begin{aligned} v_1 \le w_1 \wedge \\ (v_1 = w_1) \rightarrow v_2 \le w_2 \wedge \\ (v_1 = w_1 \wedge v_2 = w_2) \rightarrow v_3 \le w_3 \wedge \\ \ldots \end{aligned}$$

The intuition behind this definition is that if we have an assignment A, then the predicate $V \le W$ will be true of A iff $A(V) = (A(v_1), \ldots, A(v_m))$, viewed as a binary number, is less than or equal to $A(W) = (A(w_1), \ldots, A(w_m))$.

Henceforth, we fix an ordering $V = (v_1, \ldots, v_m)$ of the variables in L. Then the set of truth assignments of L inherit a lexicographic ordering, i.e., $A < B$ if, for some i, $A(v_j) = B(v_j)$ for $j < i$, while $A(v_i) < B(v_i)$.

[3] It is natural to write this as a "left action," e.g., we have ${}^{\theta\phi}A = {}^\theta({}^\phi A)$, whereas expressing the image of A under θ by A^θ would lead to the awkward relation $A^{\theta\phi} = (A^\phi)^\theta$.

[4] We note that this is surely not the *only* way to create symmetry-breaking predicates. One can break symmetries by adding any predicate that is true of one member of each equivalence class.

In other words, viewed as binary numbers, $A(V) < B(V)$.

Now consider a symmetry θ of a theory T. The predicate $V \leq V^\theta$ rules out any model M for which $M > {}^\theta M$. It is immediate that

Proposition 3.1 *Let T be a theory, and V be an ordering of it's variables. Then the predicate*

$$\bigwedge_{\theta \in \mathrm{Aut}(T)} V \leq V^\theta$$

is true only of the lexicographically least model in each equivalence class of truth assignments. Hence, it is a symmetry-breaking predicate for T.

Returning to the example we have been tracking, we take $V = (a, b, c)$. Recall that θ swaps a and b. Thus $V^\theta = (b, a, c)$, so $V \leq V^\theta$ is:

$$\begin{aligned} a &\rightarrow b \\ a = b &\rightarrow (b \rightarrow a) \\ (a = b \wedge b = a) &\rightarrow (c \rightarrow c) \end{aligned}$$

In which the only non-tautologous term is $a \rightarrow b$. This rules out the model (t, f, f). This model is ruled out because θ maps it to the lexicographically smaller model (f, t, f).

By addition of auxiliary variables the predicates given by $V \leq V^\theta$ can be represented by a linear number of clauses. We do this by introducing a new variable e_i defined to be true exactly when $v_i = v_i^\theta$. This can be done by adding the following clauses:

$$\begin{aligned} (v_i \wedge v_i^\theta) &\rightarrow e_i \\ (\overline{v_i} \wedge \overline{v_i^\theta}) &\rightarrow e_i \\ (e_i \wedge v_i) &\rightarrow v_i^\theta \\ (e_i \wedge \overline{v_i}) &\rightarrow \overline{v_i^\theta} \end{aligned}$$

Nevertheless, since $\mathrm{Aut}(T)$ may be of exponential size, the entire symmetry-breaking predicate given by this theorem may be quite large. In general we have the following negative result:

Theorem 3.2 *The problem of computing, for any theory T, a predicate true of only the lexicographic leader in each equivalence class of models is NP-hard.*

The proof of this theorem is technical and is given in the appendix. The proof includes showing the NP-completeness of the following question: Given an incidence matrix A of a graph Γ, can one reorder the vertices and edges of Γ so as to produce an incidence matrix B that exceeds A lexicographically?

Despite this negative worst-case result, it is still possible to generate, either exactly or approximately, symmetry-breaking predicates for interesting problems. In the next two sections we focus on exact methods and show that in some cases where $\mathrm{Aut}(T)$ is exponential it may still be possible to generate tractable symmetry-breaking predicates. Then, in section 6, we turn to approximate symmetry breaking.

4 The Symmetry Tree

For problems, like n-queens, with a relatively small number of symmetries we are done: one simply computes the symmetries and then calculates the predicate for each symmetry. However, many interesting problems have many symmetries, and computing the predicates for each symmetry yields unnecessary duplication. For example, if $\theta, \phi \in \mathrm{Aut}(T)$ agree on the first i variables then $P_i(V, V^\theta) = P_i(V, V^\phi)$. In order to attack problems with a large number of symmetries, we first organize the symmetries into a ***symmetry tree***, and then show how the tree can be "pruned". To describe the symmetry tree and pruning methods we need some notation for permutations.

Again, let T be a theory over L and $V = (v_1, \ldots, v_m)$ be a fixed ordering of L. We can describe a permutation of the variables by listing the image of V under the permutation. For example, the permutation taking v_1 to v_2, v_2 to v_3, and v_3 to v_1 can be written as $[v_2, v_3, v_1]$. The notation is extended to ***partial permutations***, which are 1-1 maps of initial segments of V into V, thus the partial permutation taking v_1 to v_3 and v_2 to v_1 is written $[v_3, v_1]$. Note that initial segment could be empty, giving rise to the partial permutation [ ].

For purposes of the formal development to follow, it is useful to describe these partial permutations with a standard group-theoretic construction. Let $G = \mathrm{Aut}(T)$. For $0 \leq i \leq n$, let G_i be the set of permutations in G that do not move the first i variables. That is, $G_i = \{\theta \in G \mid v_j^\theta = v_j, \text{ for } 1 \leq j \leq i\}$. Thus,

$$G = G_0 \supseteq G_1 \supseteq \cdots \supseteq G_n = 1$$

(the last being the identity subgroup). For $0 \leq i \leq n$, let $\mathcal{C}_i$ denote the set of *right cosets* of G_i in G. A right coset $C \in \mathcal{C}_i$ is a set that is of the form $G_i\theta$ for some $\theta \in G$ (note that G is the disjoint union $\bigcup_{C \in \mathcal{C}_i} C$). For purposes of this paper, one can think of a right coset as a partial permutation. For this, let $\theta \in C$, then the partial permutation of length i associated with θ

is $[v_1^\theta, \ldots, v_i^\theta]$. Note that this i-tuple is independent of the choice of $\theta \in C$.

We can now describe the structure of the symmetry tree, SB(T), for T. The root of SB(T), considered to be at level 0, is G. The set $\mathcal{C}_i$ comprises the nodes at level i. Furthermore, $C \in \mathcal{C}_i$ is a parent of $C' \in \mathcal{C}_{i+1}$ iff $C' \subseteq C$. Equivalently, in terms of partial permutations, the root is [ ] and each node $[w_1, \ldots, w_{i-1}]$ will have one child $[w_1, \ldots, w_{i-1}, x]$ for each x that is the image of v_i under a symmetry mapping V_{i-1} to $(w_1, \ldots, w_{i-1})$.

To illustrate this construction, recall the example discussed in section 2. Assume that $V = a, b, c$. There are two symmetries of this theory: the identity operation, and the exchange of a and b. The first of these takes a to itself, and the second takes a to b. There will thus be two children of the root node: $[a]$ and $[b]$. Given that a is mapped to a, b is forced to map to b, so the node $[a]$ has only the child $[a\ b]$. Similarly the node $[b]$ has only the child $[b\ a]$. The final symmetry tree is shown in figure 1.

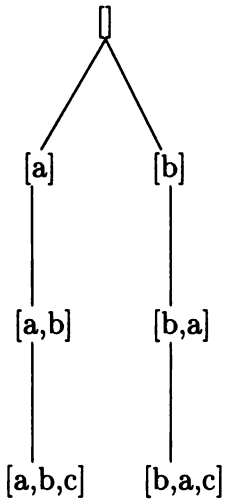


Figure 1: Symmetry trees for simple example.

The duplication previously observed in the symmetry-breaking predicate of Proposition 3.1 arose precisely because $P_i(V, V^\theta) = P_i(V, V^\phi)$ whenever θ and ϕ belong to the same right coset of G_i. With this in mind, for $C \in \mathcal{C}_i$, we define $Q(C, i)$ to be $P_i(V, V^\theta)$ for any (all) $\theta \in C$. (The "i" in the notation "$Q(C, i)$" is not superfluous, that is, it is not determined by C; it is possible that $G_i = G_j$ for $j \neq i$ in which case $C \in \mathcal{C}_j(G)$.) Finally, we associate the predicate $Q(C, i)$ to the corresponding node $C \in \mathcal{C}_i$ in SB(T). It is now clear that the conjunction of the predicates assigned to the nodes of SB(T)

$$\bigwedge_{i=1}^{n} \bigwedge_{C \in \mathcal{C}_i} Q(C, i)$$

remains a symmetry-breaking predicate for T.

5 Pruning the Symmetry Tree

Working from the symmetry tree to generate symmetry-breaking predicates eliminates a certain amount of duplication. However, there are cases in which the symmetry tree is of exponential size. For example, the theory $(x \vee y \vee z) \wedge (\overline{x} \vee \overline{y} \vee \overline{z})$ admits all $3! = 6$ permutations of $\{x, y, z\}$. Although, we do not typically expect to see all $n!$ permutations of the variables appearing in practical problems, it is not unusual to see theories with exponentially large symmetry groups. Furthermore, we would surely want to take advantage of the symmetry-breaking opportunities afforded by such a group.

In this section we show that pruning rules can achieve a drastic reduction in size of the symmetry tree in some important cases while still breaking all the symmetries. To see how this is done consider the symmetry tree shown in figure 2. Here, we suppose that Aut(T) includes the permutation $(x_1\ x_i)$, exchanging x_1 and x_i. Then, for any nodes in the symmetry of T of the form $[x_1, x_j]$ and $[x_i, x_j]$, $j \neq 1, i$, the subtree rooted at $[x_i, x_j]$ (namely, the tree T_2 in the diagram below) can be pruned.

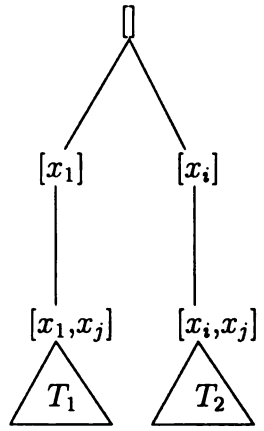


Figure 2: Symmetry trees for pruning example.

To see why this is so, consider any node $[x_i, x_j, \ldots, x_k]$ in T_2. Composing the symmetry creating this branch with $(x_1\ x_i)$, we must find a corresponding node $[x_1, x_j, \ldots, x_k]$ in T_1. In this example, it is not difficult

to show that $Q([x_1, x_j, \ldots], k) \rightarrow Q([x_i, x_j, \ldots], k)$. Thus we can prune T_2 without loss of inferential power.

The example can be generalized as follows.

Consider a right-coset $C \in \mathcal{C}_i$. Take $\theta \in C$ (the constructions to follow are independent of the choice of θ). Let $\sim_{C,i}$ be the smallest equivalence relation on L such that $v_j \sim_{C,i} v_j^\theta$ for $1 \leq j \leq i$, i.e., $v \sim_{C,i} w$ if v and w are the endpoints of a sequence $u, u^\theta, u^{\theta\theta}, u^{\theta\theta\theta}, \ldots$ such that, with the possible exception of one of these endpoints, all terms are in V_i. Observe that, if w is in neither of the sequences V_i nor V_i^θ, then the $\sim_{C,i}$ equivalence class of w is a singleton.

From the definition of Q, one can see that if $x \sim_{C,i} y$ then for any child $C', i+1$ of C, i in $SB(T)$, the antecedent of $Q(C', i+1)$ forces $x = y$. From this observation one can show:

Theorem 5.1 *Let $C \in \mathcal{C}_i$. Suppose that $\theta \in G = \mathrm{Aut}(T)$ stabilizes the equivalence classes of $\sim_{C,i}$ (i.e., $v \sim_{C,i} v^\theta$ for all $v \in L$). Then, for all $j > i$ and $C' \in \mathcal{C}_j$ with $C' \subset C$,*

$$Q(C'\theta, j) \rightarrow Q(C', j).$$

Hence, all descendents of the level i node C' in $\mathrm{SB}(T)$ *may be pruned.*

Proof: Let C', j be as indicated. Let $g \in C'$. We must show that the conjunction of the predicates

$$(V^{\phi\theta})_{j-1} = V_{j-1} \quad \rightarrow \quad v_j \leq v_j^{\phi\theta}, \tag{1}$$

$$(V^{\phi})_{j-1} \quad = \quad V_{j-1} \tag{2}$$

imply $v_j \leq v_j^\phi$.

Now, (2) implies $V_i^\phi = V_i$, which implies equality of all the variables in each the equivalence classes of $\sim_{C,i}$. But, since θ stabilizes these classes, this implies $V^\theta = V$ (i.e., the conjunction of all $v^\theta = v$), and, therefore, $V^{\phi\theta} = V^\phi$. Hence, (2) and (3) imply $v_j \leq v_j^\phi$ as required. □

In applying the condition globally, we need to be sure that we do not overprune.

It is safe to prune in all instances where $\sim_{C\theta,i} \neq \sim_{C,i}$, in which case, $\sim_{C\theta,i}$ is a proper refinement of $\sim_{C,i}$ (for C, θ, i as in the theorem, $v \sim_{C\theta,i} w$ always implies $v \sim_{C,i} w$).

In the case that $\sim_{C\theta,i} = \sim_{C,i}$, one could prune provided that $C\theta \prec C$ (there is an induced lexicographic ordering on $\mathcal{C}_i$, which is easily seen via the partial-permutation interpretation of cosets).

Although some technical extensions to this theorem are possible, this formulation is particularly useful because suitable θ can be found using standard tools of computational group theory. The computation employs "set-stabilizer" techniques that are closely related to graph-isomorphism methods. In particular, methods of Luks [1982] are guaranteed to exhibit suitable θ, if such elements exist, in polynomial time under various conditions, including boundedness of the equivalence classes of $\sim_{C,i}$. As it turns out, in practical computation, this is rarely a difficult problem anyway and is are generally considered to have efficient implementations at least for the cases corresponding to $i \leq 10000$ [Butler, 1991].

One case in which pruning is particularly effective is when the symmetry group of the theory is the full symmetric group (that is, any permutation of L is a symmetry of the theory). In this case the symmetry tree is of size $n!$, but after pruning only n^2 nodes remain. To see why this happens, note that since any perturbation is a symmetry, for any $C \in \mathcal{C}_i$ there will always be a $\theta \in \mathrm{Aut}(T)$ that stabilizes the equivalence classes of $\sim_{C,i}$. So for any node C we can always delete *all* it's descendents (so long as we have not already deleted the node $C\theta$!). If one prunes while the tree is being generated, the entire pruned tree (and thus the symmetry-breaking predicate) can be generated in n^2 time. The resultant predicate is not minimal. It turns out that if $V = (v_1, \ldots, v_m)$ then the predicate that one generates consists of a clause $v_i \rightarrow v_j$ for any i, j such that $1 \leq i < j \leq n$. There are obvious polynomial time simplifications that will reduce this to a linear number of clauses, but it is not clear how useful these simplifications will be in the general case.

Remark. Other, less extreme, cases can be constructed in which pruning is effective. However, there are also cases in which the symmetry tree is not prunable to polynomial size. The existence of such cases is a consequence (assuming P≠NP) of theorem 3.2; in fact, the theorem suggests, more strongly, that for some theories, there is no tractable lex-leader predicate since lex-leader *verification* is NP-hard. However, we can also directly construct theories where the symmetry tree cannot be pruned to polynomial size even though the lex-leader problem is in polynomial time (the algorithm uses the "string canonization" procedure of [Babai and Luks, 1983], applicable because the group turns out to be abelian). The existence of the polynomial time algorithm, in turn, guarantees we can find *some* symmetry-breaking predicate in polynomial time even though $\mathrm{SB}(T)$ is useless for this purpose. Details will appear in a future paper.

6 Approximation

If the symmetry tree is of exponential size, and no pruning is possible, then a natural approach is to generate just a part of the tree, and from this smaller tree generate partial symmetry-breaking predicates. We call a predicate P a *partial symmetry-breaking predicate* for a theory T if the models of P consist of at least one member of each of the symmetry equivalence classes of the truth assignments of the variables in T.

We can thus add P without changing the soundness or completeness of the subsequent search. The trade-off here is that the search engine may visit multiple nodes that are equivalent under some symmetry of T. In essence, then, approximating the symmetry-breaking predicate trades time spent generating symmetry-breaking predicates for time in the search engine.

In the next section we discuss various approaches to generating partial symmetry-breaking predicates.

7 Experimental Results

We have implemented a prototype system that takes a propositional theory in clausal form and constructs an approximate symmetry breaking formula from it. The implementation consists of the following steps:

1. The input theory is converted into a graph such that the automorphisms of the graph are exactly the symmetries of the theory. This is done using the construction given in [Crawford, 1992]. There are three "colors" of vertices in this graph, the vertices representing positive literals, those representing negative literals, and those representing clauses. Graph automorphisms are constrained to always map nodes to other nodes of the same color. We also add edges from each literal to each clause that it appears in. These edges (together with the node colorings) guarantee that automorphisms of the graph are symmetries of the theory.[5]

2. We find the generators of the automorphism group of the graph using McKay's graph isomorphism package, nauty [McKay, 1990]. nauty is very fast in practice though there are known examples of infinite classes of graphs which drive nauty to provably exponential behavior [Miyazaki, 1996].

3. From the generators of the automorphism group we construct the symmetry tree and then generate the symmetry-breaking predicate. As expected, in many cases computing the entire symmetry-breaking predicate is computationally infeasible. We use several approximations to compute partial symmetry-breaking predicates:

   - generating predicates for just the generators returned by nauty,
   - building the symmetry tree to some small depth and generating predicates for this smaller tree, and
   - generating random group elements and writing predicates for only those elements.

   An alternative, not yet implemented, would be to use pruning rules such as those from section 5 (though these obviously will not work in all cases).

In the experiments below we generally compare the run time for testing the satisfiability of the input theory alone and conjoined with the symmetry-breaking predicate. In all cases SAT checking was done using the TABLEAU algorithm [Crawford and Auton, 1996] and run times are "user" time. All code is written in C.

7.1 Experiment 1: The pigeonhole problem

The pigeonhole problem $PHP(n, n-1)$ is the following: place n pigeons in $n-1$ holes such that each pigeon is assigned to a hole and each hole holds at most one pigeon. This problem is obviously unsatisfiable. We study this problem because it is provably exponentially hard for any resolution based method, but is tractable using symmetries. A typical encoding of the problem is to have variables $\{P_{ij} | 1 \leq i \leq n, 1 \leq j \leq (n-1)\}$ where P_{ij} is taken to mean that pigeon i in hole j. $PHP(n, n-1)$ is then:

$$(\forall i \forall j \forall k\ (j \neq k) \Rightarrow (\overline{P_{ij}} \vee \overline{P_{ik}})) \wedge (\forall i \vee_{1 \leq j \leq (n-1)} P_{ij}) \wedge (\forall j \vee_{1 \leq i \leq n} P_{ij})$$

Since all the pigeons are interchangeable and all the holes are interchangeable, the automorphism group of $PHP(n, n-1)$ is the direct product of 2 symmetric groups. The order of this group $(n!(n-1)!)$ prohibits the full use of the symmetry tree. Furthermore, as we

[5] For efficiency we special-case binary clauses by representing $x \vee y$ with a link directly from x to y (instead of creating a node for the binary clause and linking x and y to it). This is important because some of the instances we consider have a huge number of binary clauses and some of the algorithms that follow are quadratic, or worse, in the number of nodes.

demonstrate in the Appendix (see final remark), pruning as in section 5 cannot help in this case. Hence, for these experiments, we generate only those predicates that are associated with the generators of the automorphism group (or, more specifically, the set of generators returned by nauty). For PHP, such generators are not only determined in polynomial time, but also serve to break all symmetries. (Of course, in general, the predicates associated with generators of $\mathrm{Aut}(T)$ do not suffice to break all symmetries.)

The run times for various sizes of the n are shown in figure 3. Run times are on a Sparc 10:51

It is difficult to tell from the run-time data what the scaling is, but it turns out that we can show analytically that every step of our implementation is in polynomial time. The input theory can be represented as a graph with $3n^2 - 2n + 2$ vertices[6]. nauty takes this graph as input and finds the generators of its automorphism group. To do this nauty builds a search tree in which each node is a coloring of the vertices which is a suitable refinement of the coloring of the parent node.[7] The time that nauty spends on each node is polynomial in the size of the input graph. So to show that nauty runs in polynomial time it suffices to show that the number of nodes is polynomial. The proof requires a discussion of the details of the internals of nauty that is beyond the scope of this paper, but one can show that for this problem nauty expands exactly $2n^2 - 3n - 1$ nodes. Computing symmetry-breaking predicates for the generators is obviously in polynomial time. The last step is SAT checking which is, in general, exponential, but for these theories, augmented with the symmetry-breaking predicates, there was no need to run TABLEAU: a proof of unsatisfiability was obtained by a polynomial time simplification procedure that is used as a front-end to TABLEAU.

7.2 Example 2: N-queens

The n-queens problem has been well studied in the CSP literature, but we include it here as a prototypical example of a problem with a small number of geometric symmetries. The problem is to place n queens on a n by n chess board such that no two queen can attach each other. N-queens has 8 symmetries and for any size board the full symmetry tree has eight leaves.

As one can see from the construction in section 4,

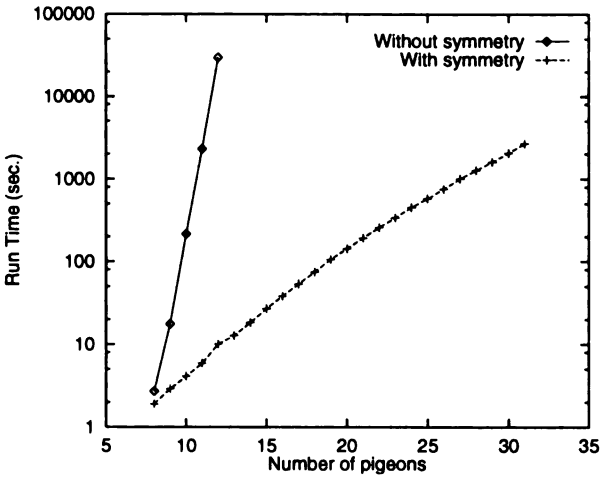


Figure 3: Run times for the pigeon-hole problem with and without symmetry. Note that the y axis is log scaled.

nodes at depth i in the symmetry tree generate clauses of length linear in i. It turns out that long clauses are of very little use to a satisfiability checker like TABLEAU, so for these experiments we cut off the symmetry tree at depth 20 and generate predicates only up to this depth. The results are shown in figure 4. Run times are for a Sparc 5.

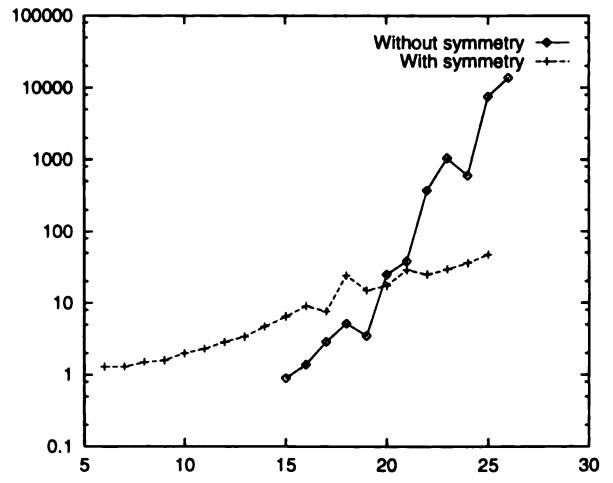


Figure 4: Run times for n-queens with and without symmetry. Note that the y-axis is log scaled.

We know that N-queens is a somewhat delicate problem in that reordering the clauses in the input can drastically change the behavior of SAT checkers (especially as n is increased). Therefore, we took each theory (with and without symmetry-breaking predicates)

[6]The theory actually has $O(n^3)$ clauses, but many of these clauses are binary clauses that become edges in the graph rather than nodes

[7]A refining of a vertex coloring C is another vertex coloring $\hat{C}$ such that if vertex i and j have the same color in $\hat{C}$ then they have the same color in C.

and randomly perturbed the order of the clauses (and of the variables within the clauses) 50 times.[8][9] We then ran TABLEAU on each permuted theory. Figure 5 shows the average run times. As can be seen, the qualitative nature of the results has not changed but a fair amount of noise has been removed.

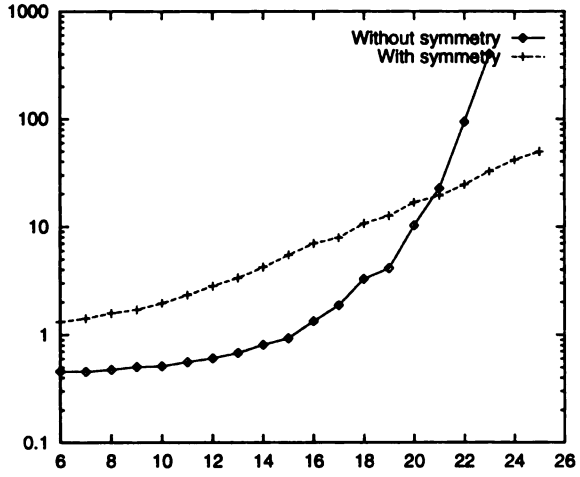


Figure 5: Average run times over 50 random permutations with and without symmetry. Note that the y-axis is log scaled.

8 Related Work

[Freuder, 1991] discusses the elimination of interchangeable values in constraint satisfaction problems. [Brown *et al.*, 1988] discuss an algorithm for backtracking search in the presence of symmetry. In their approach the search engine is modified so that at each node in the search tree a test is done to determine whether the node is lex-least under the symmetries of the theory. [Krishnamurthy, 1985] discusses the idea of using symmetries to reduce the lengths of resolution proofs. He uses the "rule of symmetry" which asserts that if θ is symmetry of a theory Th, and one can show that p follows from Th, then $\theta(p)$ also follows from Th (since the proof could be repeated with each step s replaced by $\theta(s)$). Tour and Demri [1995] show that Krishnamurthy's method is NP-complete in general, but that a restriction of it is equivalent to graph isomorphism. Their restricted method appears to be the same as that discussed by Crawford [Crawford, 1992], and Tour and Demri's proof of graph isomorphism uses essentially the same construction used by Crawford. Benhamour and Sais [1992] also discuss techniques for making use of symmetries within specially designed search engines. The most successful uses of symmetry in reducing search spaces is surely in the large and growing literature on the application of automorphism groups to combinatorial problems (see, *e.g.*, [Butler and Lam, 1985; Lam and Thiel, 1989; Lam, 1993; Laue, 1995]). This work has made impressive contributions to the discovery and classification of designs and to the study of combinatorial optimization problems.

[8] Obviously these perturbations have nothing to do with symmetries of the theory. The idea here is only to average out the cases where the SAT checker gets "lucky" and stumbles on a model almost immediately.

[9] For 23 queens the data given is for 20 perturbations.

9 Concluding remarks

This work has shown how symmetries can be utilized to add additional constraints, symmetry-breaking predicates, to search problems. These constraints ensure that the search engine never visits two points in the search space that are equivalent under some symmetry of the problem. Complexity results suggest that generating symmetry-breaking predicates will be intractable in the general case. However, partial symmetry breaking can be done in polynomial time (assuming the associated graph isomorphism problem is tractable). Preliminary experiments have been completed showing that partial symmetry breaking is effective on prototypical constraint-satisfaction problems.

Appendix

Theorem 3.2 is a direct consequence of the NP-completeness of

Problem. MAXIMUM IN MODEL CLASS (MMC)

Input: A theory T over L; an ordering of L; $M \in \mathcal{M}(T)$.

Question: Does there exist $\theta \in \mathrm{Aut}(T)$ such that ${}^{\theta}M < M$?

We shall demonstrate completeness by a reduction from CLIQUE [Garey and Johnson, 1979]. We make use of an intermediate problem, which is interesting in its own right. For lexicographic ordering, we consider $m \times n$ matrices, as mn-tuples, taking the rows in succession.

Problem. MAXIMUM INCIDENCE MATRIX. (MIM)

Input: An incidence matrix A for a graph Γ.

Question: Is there an incidence matrix B for Γ such that $B > A$ with respect to lexicographic ordering.

Recall that a $|V| \times |E|$ incidence matrix of $\Gamma = (V, E)$ is determined by specified orderings of V and E, wherein $A_{ij} = 1$ or 0 according to whether or not the ith vertex lies on the jth edge. Two such matrices are then related by permutations of the rows and columns. Thus, in particular, the NP-completeness of MIM establishes the NP-completeness of the problem of the existence of a matrix B, obtained from a given $\{0,1\}$-matrix A by permuting rows and columns, such that $B > A$.

Lemma 9.1 *MIM is NP-complete.*

Proof: MIM is in NP since suitable orderings of V and E can be guessed and verified. For the completeness, we reduce CLIQUE to MIM. Suppose we are given an instance (Γ, K) of CLIQUE, wherein $\Gamma = (V, E)$ is a graph and K is a positive integer; the relevant question is whether Γ contains a K-clique (i.e., a complete subgraph on K vertices). We may assume $|V| > K > 3$.

We augment Γ to a graph $\hat{\Gamma} = (\hat{V}, \hat{E})$ as follows. Fix an ordering $v_1, v_2, \ldots, v_{|V|}$ of V. The additional vertices comprise sets W, X, and Y, disjoint from V and from one another. We describe these along with new edges:

1. W is a complete graph on K vertices, $\{w_1, w_2, \ldots, w_K\}$, ordered as indicated by the subscripts.
2. For each $w_i \in W$, join w_i to each vertex in a new set X_i of size $|V|(|V|+1)/2)$; $X = \bigcup_i X_i$ is ordered so that X_i precedes X_j for $i < j$. Only X_1 is joined to V and this is done by joining the first $|V|$ elements of X_1 to v_1, the next $|V|-1$ elements to v_2, the next $|V|-2$ to v_3, etc.
3. The set Y, joined only to V, provides a new common neighbor for each pair of vertices V and also ensures the degree of each $v \in V$ is maximal in $\hat{\Gamma}$. Thus, for $1 \leq i < j \leq |V|$, we create a new vertex y_{ij} and add edges joining it to v_i and v_j. Finally, for each $v_i \in V$, join v_i to the vertices in a new set Y_i chosen so as to bring the total degree of v_i to precisely $d = K - 1 + |V|(|V|+1)/2$ (which is also the degree of each $w \in W$). Let $Y = \{ y_{ij} \mid 1 \leq i < j \leq |V| \} \cup \bigcup_i Y_i$. We order Y so that: for $i < j$, y_{ij} precedes Y_i; for $i < i' < j$, Y_i precedes $y_{i'j}$; and for $i < j < j'$, y_{ij} precedes $y_{ij'}$.

The vertices $\hat{V} = W \cup X \cup V \cup Y$ in the resulting graph $\hat{\Gamma}$ are ordered in the sequence $WXVY$ with the orders within each of the four segments as indicated above.

By construction, $\hat{\Gamma}$ has a K-clique, W. Observe, however, that $\hat{\Gamma}$ has a *second* K-clique iff Γ had a K-clique, since the vertices in $X \cup Y$ have degree at most 2.

For the instance of MIM, we take A to be the lexicographically greatest incidence matrix of $\hat{\Gamma}$ with respect to the indicated ordering of the vertices. In this regard, note that the maximum incidence matrix for a *given* vertex ordering is obtainable in polynomial time.

We claim that A is the maximum incidence matrix for $\hat{\Gamma}$ if W is first in the ordering of $\hat{V}$. For, suppose A' is the maximum such matrix. Then rows $K+1$ through $K+|X|$ of A' must again correspond to X, the only other vertices directly joined to W; which forces these rows to duplicate their counterparts in A. The next $|V|$ rows in A' must now correspond to V, since these are the only remaining vertices directly joined to Y; suppose these rows correspond to the ordering $v_{i_1}, v_{i_2}, v_{i_3}, \ldots$ of V. Since the v_{i_1} row in A' cannot be exceeded by the v_1 row in A, v_{i_1} must also be joined to $|V|$ elements of X, in fact, to the first $|V|$ elements; so $i_1 = 1$. Similarly, $i_2 = 2$, $i_3 = 3$, etc., so that V remains ordered as before. But, given this ordering of V, the maximality of A' requires Y to be ordered as specified above (except for irrelevant reorderings within each Y_i). Thus, $A' = A$, proving the claim.

We need to show that $\hat{\Gamma}$ has a second K-clique iff $\hat{\Gamma}$ has an incidence matrix $B > A$.

Suppose $\hat{\Gamma}$ has a K-clique $W' \subseteq V$. Reorder $\hat{V}$ so that W' comprises the first K elements and the $(K+1)$st element is the common neighbor in Y of the first two elements of W'. The matrix B induced by the new order agrees with A in the first K rows but its $(K+1)$st row exceeds that of A (the $(K+1)$st row of A begins $0^{K-1}10^{d-2}0$, while the $(K+1)$st row of B begins $0^{K-1}10^{d-2}1$). Hence $B > A$.

Conversely, let B be the maximum incidence matrix for $\hat{\Gamma}$ and suppose that $B > A$. Since the first K rows of A recorded a K-clique consisting entirely of vertices of maximum degree d in $\hat{\Gamma}$, this segment cannot be strictly exceeded. Hence, the first K vertices in the ordering that produces B must form a clique. Since $B > A$, this clique is not W. □

Remark. We trust that the above demonstration will discourage finding-lex-leading-incidence-matrices as an approach to finding canonical forms for graphs and, thereby, to graph isomorphism.

Lemma 9.2 *MMC is NP-complete.*

Proof: MMC is in NP since, one can guess θ, if it

exists, and verify both $\theta \in \mathrm{Aut}(T)$ and ${}^{\theta}M < M$ in polynomial time. We reduce MIM to MMC as follows. Suppose the $m \times n$ matrix A constitutes an instance of MIM. Let X and Y be sets of size m, n respectively We describe a theory T on the set $L = X \times Y$ of variables, namely,

$$T = \bigwedge_{\substack{x,x' \in X \\ y \in Y}} \left((x,y) \vee (x',y) \vee \overline{(x',y)}\right) \quad \wedge \quad \bigwedge_{\substack{x \in X \\ y,y' \in Y}} \left(\overline{(x,y)} \vee (x,y') \vee \overline{(x,y')}\right)$$

Trivially, T is a tautology. One verifies that $\mathrm{Aut}(T)$ is $\mathrm{Sym}(X) \times \mathrm{Sym}(Y)$ acting on $X \times Y$ in the natural way. We fix orderings $x_1, x_2, \ldots, x_m$ and $y_1, y_2, \ldots, y_n$ of X and Y respectively, and let $X \times Y$ be ordered lexicographically. The $m \times n$ $\{0,1\}$-matrix A yields a model M of T wherein $M((x_i, y_j)) = 1 - A_{ij}$ (the $0,1$ reversal to accommodate a conversion from maximal matrices to minimal models). There is a natural correspondence between row (respectively, column) permutations and $\mathrm{Sym}(X)$ (respectively, $\mathrm{Sym}(Y)$). Thus, if B is obtained from A via a row permutation and a column permutation, then the permutation pair yield $\theta \in \mathrm{Sym}(X) \times \mathrm{Sym}(Y)$. It follows directly that $B > A$ iff ${}^{\theta}M < M$, establishing the reduction of MIM to MMC. □

Remark. This particular proof of the NP-completeness of MMC was chosen so as to show that the problem remains NP-complete even when L can be identified with some $X \times Y$ and $\mathrm{Aut}(T) = \mathrm{Sym}(X) \times \mathrm{Sym}(Y)$. Note that the pigeonhole problem engenders a theory of exactly this type. (the fact that $|X| = |Y| - 1$ in PHP is not significant - routine padding could be used to force this restriction in MIM). Thus, we have demonstrated the futility, for PHP, of pruning via methods, like that of section 5, which rely solely on information contained in $\mathrm{Aut}(T)$ and $\mathrm{SB}(T)$.

Acknowledgements

This work was supported in part by ARPA/Rome Labs under grant numbers F30602-93-C-0031 and F30602-95-1-0023, by AFOSR under grant numbers F49620-92-J-0384 and F49620-96-1-0335, and by NSF under grant number IRI-94 12205. The work benefited from various discussions over the years with many people including Bart Selman, Steve Minton, Takunari Miyazaki, David Etherington, David Joslin, and all the members of CIRL.

References

[Babai and Kučera, 1979] L. Babai and L. Kučera. Canonical labelling of graphs in linear average time. In *Proceedings of the Twentieth IEEE Conference on Foundations of Computer Science*, pages 46–49, 1979.

[Babai and Luks, 1983] László Babai and Eugene M. Luks. Canonical labeling of graphs. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, pages 171–183, Boston, Massachusetts, 25–27 April 1983.

[Babai, 1995] L. Babai. Automorphism groups, isomorphism, reconstruction. In L. Lovász R. L. Graham, M. Grötschel, editor, *Handbook of Combinatorics*, chapter 27, pages 1447–1540. North-Holland - Elsevier, 1995, 1995.

[Benhamou and Sais, 1992] Belaid Benhamou and Lakhdar Sais. Theoretical study of symmetries in propositional calculus and applications. In D. Kapur, editor, *Automated Deduction: 11th International Conference on Automated Deduction (CADE-11)*, Lecture Notes in Artificial Intelligence, pages 281–294. Springer-Verlag, 1992.

[Brown *et al.*, 1988] Cynthia A. Brown, Larry Finkelstein, and Paul W. Purdom. Backtrack searching in the presence of symmetry. In T. Mora, editor, *Applied algebra, algebraic algorithms and error correcting codes, 6th international conference*, pages 99–110. Springer-Verlag, 1988.

[Butler and Lam, 1985] G. Butler and C. W. H. Lam. A general backtrack algorithm for the isomorphism problem of combinatorial objects. *Journal of Symbolic Computation*, 1(4):363–382, 1985.

[Butler, 1991] G. Butler. *Fundamental algorithms for permutation groups*. Lecture notes in computer science. Springer-Verlag, 1991.

[Crawford and Auton, 1996] James Crawford and Larry Auton. Experimental results on the crossover point in random 3sat. *Artificial Intelligence*, 81, 1996.

[Crawford, 1992] James Crawford. A theoretical analysis of reasoning by symmetry in first-order logic (extended abstract). In *Workshop notes, AAAI-92 workshop on tractable reasoning*, pages 17–22, 1992.

[de la Tour and Demri, 1995] Thierry Boy de la Tour and Stéphane Demri. On the complexity of extending ground resolution with symmetry rules. In *Pro-*

ceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95), volume 1, pages 289–295, 1995.

[Freuder, 1991] Eugene G. Freuder. Eliminating interchangeable values in constraint satisfaction problems. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 227–233, 1991.

[Garey and Johnson, 1979] M.R. Garey and D.S. Johnson. *Computers and Intractability.* W.H. Freeman and Co., New York, 1979.

[Gaschnig, 1979] John Gaschnig. Performance measurement and analysis of certain search algorithms. Technical Report CMU-CS-79-124, Carnegie-Mellon University, 1979.

[Ginsberg, 1993] Matthew L. Ginsberg. Dynamic backtracking. *Journal of Artificial Intelligence Research*, 1:25–46, 1993.

[Krishnamurthy, 1985] B. Krishnamurthy. Short proofs for tricky formulas. *Acta Informatica*, 22:253–275, 1985.

[Lam and Thiel, 1989] C. W. H. Lam and L. Thiel. Backtrack search with isomorph rejection and consistency check. *Journal of Symbolic Computation*, 7(5):473–486, 1989.

[Lam, 1993] C. W. H. Lam. Applications of group theory to combinatorial searches. In L. Finkelstein and W. M. Kantor, editors, *Groups and Computation, Workshop on Groups and Computation*, volume 11 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 133–138, 1993.

[Laue, 1995] R. Laue. Construction of groups and the constructive approach to group actions. In S. Walcerz T. Lulek, W. Florek, editor, *Symmetry and Structural Properties of Condensed Matter in Proceedings of the Third International School on Theoretical Physics*, pages 404–416. World Scientific - Singapore, New Jersey, London, Hong Kong, 1995.

[Luks, 1982] E. M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *J. Comp. Sys. Sci.*, 25:42–65, 1982.

[McKay, 1990] B. D. McKay. *Nauty user's* guide, version 1.5. Technical Report TR-CS-90-02, Department of Computer Science, Australian National University, Canberra, 1990.

[Minton *et al.*, 1990] Steven Minton, Mark D. Johnston, Andrew B. Philips, and Philip Laird. Solving large-scale constraint satisfaction and scheduling problems using a heuristic repair method. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 17–24, 1990.

[Miyazaki, 1996] Takunari Miyazaki. The complexity of McKay's canonical labeling algorithm. In L. Finkelstein and W. M. Kantor, editors, *Groups and Computation II, Workshop on Groups and Computation*, volume to appear of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, 1996.

[Selman *et al.*, 1992] Bart Selman, Hector Levesque, and David Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 440–446, 1992.

Procedural Reasoning in Constraint Satisfaction

Ari K. Jónsson
Computer Science Department
Stanford University
Stanford, CA 94305

Matthew L. Ginsberg
CIRL
1269 University of Oregon
Eugene, OR 97403

Abstract

For many constraint satisfaction problems, there are well known, fast algorithms and functions that solve parts of the problem. Using these methods directly to solve the subproblems significantly speeds up solving process. Unfortunately, doing this has usually required the solver to be changed, or the correctness criteria to be re-examined. We describe a general mechanism to use procedures with almost any search engine, such that it is easy to add any procedures without changing the engine. Furthermore, the framework is formally defined, which allows us to prove conditions that are sufficient to guarantee systematicity and completeness for search engines using procedures.

1 Introduction

For many constraint satisfaction problems there are simple functional relations (e.g. arithmetic equations) and simple subproblems (e.g. linear equations with unknowns) that can be solved quickly, using simple algorithms. Needless to say, taking advantage of such algorithms can significantly decrease the time needed to find a solution. But doing so without rewriting solvers and redoing correctness proofs, has proved to be difficult.

To demonstrate this, let us look at a small constraint satisfaction problem that we will use throughout this paper. Assume that we are finishing a shift in an automated workshop, and we have two unfinished objects. A chair needs to be sanded and then painted, and a table needs to be painted. We have two robots, one that sands and another that paints. Obviously the same robot cannot do two jobs at the same time and the same object cannot be sanded and painted at the same time. Each job takes one hour, and we need to be done in two hours.

The constraint satisfaction problem will determine when each robot should start each of its jobs, but it is well known in the operations research community that the real decisions to be made are which job is done before which other job.[1] These observations lead to the following specification (which is a simplification of an encoding used by others [Crawford and Baker, 1994]):

The variables and possible values are:

```
(pc->pt) : {true,false}
(pt->pc) : {true,false}
(pc->sc) : {true,false}
(sc->pc) : {true,false}
(pt->sc) : {true,false}
(sc->pt) : {true,false}
start(sc) : {0,1,2}
start(pt) : {0,1,2}
start(pc) : {0,1,2}
```

where `pc` means "paint chair", `pt` means "paint table", `sc` means "sand chair", and the arrow means "precedes". The constraints are:

1. `(sc->pc) = true`
2. `(pc->pt) = true` or `(pt->pc) = true`
3. if `(X->Y)=true` and `start(X)=k`, then `start(Y)>k`
4. `start(X) < 2`

where `X` and `Y` range over the jobs to be done. Constraint 1 states that the chair is sanded before it is

[1] Smith and Cheng introduced this fact to the AI community [Smith and Cheng, 1993].

painted. Constraint 2 states that the two objects cannot be painted simultaneously. Constraints 3 says that a job cannot start until all jobs that come before it, have finished. Finally constraint 4 guarantees that the schedule is at most two hours long.

The problem is easy to solve with search, and we get `start(sc)=0`, `start(pt)=0` and `start(pc)=1`. But for this small problem, like any constraint satisfaction problem, there are polynomial algorithms that solve certain subproblems.

First, there is unit propagation, which is a well-known algorithm in the CSP community. The idea behind it is simple: If there is a constraint where all but one of the variables that appear in it have been assigned values, and the last unassigned variable can only be assigned one specific value to satisfy the constraint, assign that value to the unassigned variable. In this example, if `(pt->pc)=false` then we have to assign the value `true` to the variable `(pc->pt)` to satisfy constraint 2.

Second, there is a well known operations research algorithm, often referred to as PERT scheduling [Hillier and Lieberman, 1986]. This algorithm calculates the earliest starting time and latest finishing time for each task in a partially ordered schedule. This algorithm can easily be used here to find the earliest possible starting times for each job that hasn't been assigned a starting time yet, and thus check if the schedule may still fit within the time limit. Note that we cannot directly assign the earliest starting times found by PERT, since a given starting time may satisfy the partial order, but violate some other constraints. In this case we will use the algorithm as a pruning technique: Given a partial ordering of the jobs, i.e. assignments to some of the `(X->Y)` variables, our simplified version calculates the earliest possible starting time for each job and checks if the given partial ordering will fit within the time limit. As an example, if we assign `(sc->pc)=true` and `(pc->pt)=true`, then the algorithm returns failure, since the job `pt` cannot start earlier than time 2, violating constraint 4. As another example, assume we have a cycle in the schedule, e.g. `(sc->pc)=true` and `(pc->sc)=true`. The simplified PERT algorithm will then return failure, since this assignment violates constraint 3.

When confronted with such a scheduling problem, we want to take advantage of these two algorithms to reduce the amount of search done. At the same time, we want to use the best available search engine to minimize the time spent searching. To do this, we need a general method to add such algorithms to search engines. Using earlier techniques, there are three possible approaches. The first possibility is to use a general purpose search engine with a declarative description of the procedures. The second option is to use a description of a good search engine, and write a specific implementation of it, which uses the procedures above. Finally, we can have a search engine that uses procedural attachments [Green, 1969] [2] 8to evaluate certain variables, and then add unit propagation and PERT as attachments. Unfortunately, none of these solutions is satisfactory.

A general search engine is a valuable tool in itself, but almost no value is added by using declarative descriptions for procedural reasoning methods, since such descriptions invariably become a part of the CSP that is to be solved. The problems this causes are well known.

- Although adding declarative knowledge to the CSP may in some cases speed up the search effort, the side effect is increased problem size, often leading to increased solving time.
- Using declarative descriptions for simple calculations like arithmetic is much less efficient than doing the same calculations directly.
- Finding a usable description of the more complex procedures like PERT scheduling is nontrivial, and in some cases practically impossible.

The fact is that translating a procedure into a declarative description may take significant amount of work, and in many cases the rewards can be negative.

The next possibility, rewriting the solver each time we have a new technique to add, has many obvious drawbacks.

- The time and effort involved in writing a new implementation for a search engine, in particular when procedures of various types are incorporated, is significant. Applying new search techniques and comparing different search techniques for a particular problem domain becomes almost impossible without sacrificing the efficiency of applicable procedures.
- Specialized implementations tend to be brittle, so that small changes in the problem or additional constraint types will in most cases break them. In an automated reasoning system, flexibility is a crucial attribute.
- Without theoretical analysis, a search engine incorporating a number of additional algorithms

[2] A recent article provides a good description of procedural attachments [Myers, 1994].

may not have the properties that were expected. For example, some search engines that use the pure literal rule become incomplete when combined with a symmetry breaking algorithm [Crawford *et al.*, 1996].

The only advantage of specialized implementations is that clever coding can minimize the overhead of using reasoning procedures, such as unit propagation, by using data structures that are as efficient as possible for both solver and procedure.

Finally, using procedural attachments has its problems, too:

- Procedural attachments directly calculate values for problem variables, making it difficult to reuse a procedure in another search engine that has different data structures.
- The procedural attachments are basically functions, specifying the value of one variable based on assignments to other variables. This makes it practically impossible to implement global algorithms like unit propagation.
- Procedural attachments do not take into account the effects they have on the search engine. Many authors assume the search engine is cooperative, while others assume the search engine does simple backtracking [Maluszynski *et al.*, 1993]. For instance, procedural attachments do not work with some dependency directed methods.

The obvious solution to our problems is a framework where search engines and procedures can be developed and implemented separately, but used together in any combination. It is easy enough to define such a framework, e.g., define object classes for search engines and procedures, and specify a set of interface calls. This is not a real solution however, because two important issues have been disregarded. One is the question of generality: Will an arbitrarily designed object class system cover all search engines and procedures, some of which have not yet been developed? The other issue is that of correctness. There are no guarantees that each search engine will work correctly with each procedure. As an example, a search engine may eliminate one of two possible solutions using the pure literal rule, while a procedure eliminates the other solution based on the two solutions being symmetric. Separately, both objects are reasonable designs, and work with most other engines and procedures, but together they will return the wrong answer.

A satisfactory solution to this problem must be a well defined framework, based on general definitions for search engines and procedures, along with a clear specification of how the two interface. It must allow procedures and search engines to be developed and implemented separately, while making it easy to use any applicable procedures with the search engine of choice. The framework will also have to be formally defined, so that it can be theoretically proven which conditions on the procedures and search engines are sufficient to guarantee correctness, completeness and systematicity.

In this paper we present such a framework. In section 2 we present a formal description of search engines and extension procedures, and describe how the two are combined. In section 3, we prove easily satisfied conditions to be sufficient to guarantee systematicity and completeness. Concluding remarks are in section 4. For clarity, all proofs have been removed to the appendix.

2 Framework

2.1 Constraint Satisfaction Problems

A constraint satisfaction problem is a finite set of variables, a finite set of domain values for each variable, and a collection of constraints on the values assigned to the variables. This is formalized as follows:

Definition 2.1 *A* constraint satisfaction problem (CSP) *is a triple* (X, V, K), *where* $X = \{x_1, \ldots, x_n\}$ *is a finite set of variables,* $V = \{V_{x_i}\}$ *is a set of domains for each variable, and* K *is a set of constraints. Each constraint is of the form* (Y, R), *where* $Y = \{x_{i_1}, \ldots, x_{i_k}\} \subseteq X$ *and* $R \subseteq \prod_{\nu=1}^{k} V_{x_{i_\nu}}$.

To solve a constraint satisfaction problem, we need to find an assignment for each variable such that all the constraints are satisfied. Formally:

Definition 2.2 *A* solution *to a* CSP (X, V, K) *is an* n*-tuple* $(v_{x_1}, \ldots, v_{x_n})$, *where* $n = |X|$, *such that:*

1. $v_{x_k} \in V_{x_k}$ *for* $k = 1, \ldots, n$, *and*
2. *For any* $(Y, R) \in K$ *with* $Y = \{x_{i_1}, \ldots, x_{i_k}\}$, *we have* $(v_{i_1}, \ldots, v_{i_k}) \in R$.

Few methods for solving constraint satisfaction problems are able to jump directly to a solution; most work with intermediate assignments that either do not assign values to all the variables, or do not satisfy all of the applicable constraints. These intermediate assignments are referred to as partial assignments:

Definition 2.3 *Given a* CSP (X, V, K) *with n variables, a* partial assignment *is a pair of m-tuples* $(x_{i_1}, \ldots, x_{i_m})$ *and* $(v_{i_1}, \ldots, v_{i_m})$, *where* $m \leq n$ *and* $v_{i_k} \in V_{x_{i_k}}$ *for* $k = 1, \ldots, m$. *A partial assignment will be called* consistent *if for any* $(Y, R) \in K$ *with* $Y = \{x_{j_1}, \ldots, x_{j_k}\} \subseteq \{x_{i_1}, \ldots, x_{i_m}\}$, *we have* $(v_{j_1}, \ldots, v_{j_k}) \in R$. *A partial assignment that is not consistent will be called* inconsistent.

If $p = \langle Y, W \rangle$ *is a partial assignment such that* $x_i \notin Y$, *we will call the partial assignment* $\langle Y \cup \{x_i\}, W \cup \{v\} \rangle$ *the result of* extending p by assigning the value v to the variable x_i. *We will denote this by* $p + \langle x_i, v \rangle$.

2.2 Search Engines

Constraint satisfaction problems are invariably solved with a search engine, an algorithm that searches for a solution by modifying a partial assignment. The prototypical example of a search engine is the depth-first search method:

Algorithm 2.4 To SOLVE a CSP (X, V, K), given a partial assignment p:

1 **if** p is inconsistent, **return** failure
2 **if** p is a solution, **return** p
3 **let** x be a variable not assigned a value by p,
  let $E_x = V_x$
4 **if** E_x is empty, **return** failure,
  else select $v \in E_x$
5 **if** SOLVE$(p + \langle x, v \rangle)$ succeeds, **return** result,
  else remove v from E_x and **goto** 4

Taking p to initially be the empty assignment, it is well known that the algorithm is systematic and complete.

contains many such specific algorithms, but there is little discussion about search engines in abstract terms. Most researchers have focused on specific algorithms or presented ideas that intuitively apply to a certain class of algorithms. As our goal is to provide a framework into which any search engine can fit, we need a general definition for a search engine.

To see how we should define a general search engine, let us examine how the depth-first algorithm above works. We note that the E_x sets are used to control the search by keeping track of which value assignments have been tried. The algorithm makes steady progress by gradually shrinking the set of partial assignments that are consistent with the E_x sets. The recursive nature of the algorithm controls the backtracking by automatically returning to the calling function when a SOLVE function call fails. To sum up these observations, the E_x sets define the state, and the recursive calls determine how the states are updated.

Given this, the obvious way to define a search engine is to use a general state and an update function. This turns out to be too strong, since a Turing machine can be implemented within those parameters. The solution is to limit the number of possible states, but it must be done carefully. The most obvious solution, to limit the size of each state to be polynomial in the number of variables, is unsuitable as this excludes many dependency directed search methods. To see how we should limit and represent each state, let us focus on what differentiates search engines from other algorithms.

Consider any search engine that works by examining partial assignments, changing or extending them, trying to get to a solution. Such an engine will invariably have a set of partial assignments that may be considered in the near future, and will use this set to determine which partial assignment it will examine next. This is true for engines ranging from simple depth-first search and ISAMP [Langley, 1992] to dependency directed backtracking [Stallman and Sussman, 1977] and WSAT [Selman *et al.*, 1993]. Based on these observations, we define a search engine as having a state that consists of a set of partial assignments, and a step function to update its state.

To formally define a search engine, we will need to identify a few simple concepts in constraint satisfaction. Given a CSP C, we will use $\mathcal{P}_C$ to denote the set of all possible partial assignments, consistent and inconsistent. The set of all possible states, denoted by $\mathcal{S}_C$, is simply $2^{\mathcal{P}_C}$, the set of all possible sets of partial assignments. Finally, to identify partial assignments that are solutions, we will use Γ_C to denote the set of all solutions to C. Formally:

Definition 2.5 *Given a* CSP C, *let* $\mathcal{P}_C$ *be the set of all partial assignments. Let* $\mathcal{S}_C = 2^{\mathcal{P}_C}$. *Let* Γ_C *be the set of all solutions to* C.

A search engine will have a state that is a member of $\mathcal{S}_C$, i.e. a set of partial assignments. The states will be updated using a successor function `succ` that maps one state to another. Associated with each state S will be a specific partial assignment that corresponds to the partial assignment p in the depth first search engine above. This "current assignment" will be denoted by `curr`(S). Both `succ` and `curr` will be partial functions, since a given search engine will only use a small subset of $\mathcal{S}_C$ to represent all of its states. Finally, a search engine will be problem-independent, so each of the functions `succ` and `curr` take the problem C as an argument. But since a search engine is only solving one problem at a time, we will omit the CSP

parameter C for the sake of clarity.

Formally:

Definition 2.6 *A* search engine *is defined by a pair of partial functions* $(\mathtt{succ}, \mathtt{curr})$, *such that* $\mathtt{succ} : \mathcal{S}_C \to \mathcal{S}_C$, *and* $\mathtt{curr} : \mathcal{S}_C \to \mathcal{P}_C$. *Furthermore the partial functions satisfy:*

1. $\mathtt{succ}(\mathcal{P}_C)$ *is defined.*
2. $\mathtt{succ}(\emptyset) = \emptyset$.
3. *If* $\mathtt{succ}(S)$ *is defined,* $\mathtt{curr}(S)$ *is defined.*
4. *If* $\mathtt{succ}(S)$ *is defined,* $\mathtt{succ}(\mathtt{succ}(S))$ *is defined.*

This may look complicated, but the basic idea is simple. At each point, a search engine will have a state, consisting of a set of partial assignments that it views as candidates for examination. The function succ does one step in the search, updating the state by refining or changing the set of candidates. The function curr returns the partial assignment associated with that state, the "current" partial assignment. Given this, it is easy to see how (succ, curr) would typically be used to solve a CSP C:

Algorithm 2.7 To solve a CSP C, given a search engine (succ, curr):

1 **let** $S = \mathcal{P}_C$
2 **if** $S = \emptyset$, **return** failure
3 **if** $\mathtt{curr}(S) \in \Gamma_C$, **return** $\mathtt{curr}(S)$
4 **let** $S = \mathtt{succ}(S)$, **goto** 2

Conditions 1 and 2 in the definition merely state that the search engine has a zero-information initial state $\mathcal{P}_C$, and a final state $\emptyset$ that is used to terminate the search engine. Condition 3 means there is a "current" assignment associated with each state, and finally condition 4 says that once running, the search engine can keep running.

To further illustrate how this formalization works, let us look at depth-first search again. The function $\mathtt{curr}(S)$ returns a maximal-depth member of S that has no parents in S. The update function $\mathtt{succ}(S)$ returns $S - \{\mathtt{curr}(S)\}$. See Figure 1 for a small example of how the formalization of depth-first search works.

The two properties of a search engine that we are most concerned about are systematicity and completeness. Systematicity guarantees that a search engine does not get stuck, while completeness guarantees a solution will eventually be found if one exists. These properties are easy to define in this setting:

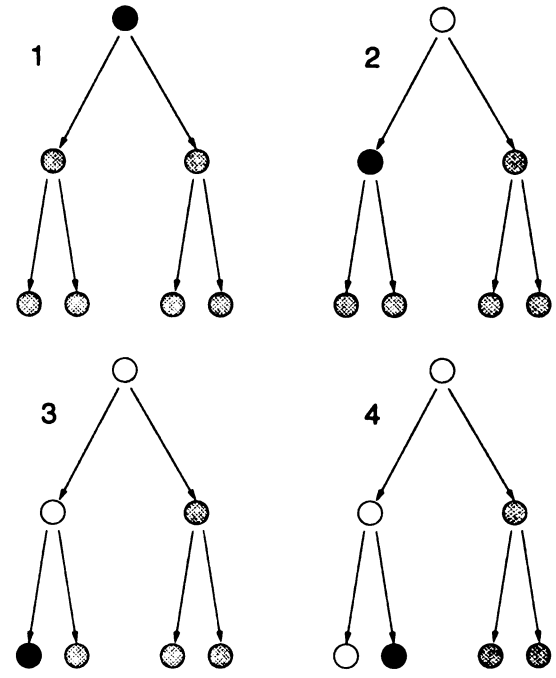


Figure 1: Depth first search, first four steps. The non-white nodes are the partial assignments in S, the black node is $\mathtt{curr}(S)$.

Definition 2.8 *A search engine* $(\mathtt{succ}, \mathtt{curr})$ *is* complete *if, for any* CSP C *such that* $\Gamma_C \neq \emptyset$, *there exists* $s \in \Gamma_C$ *and* $k \in \mathbb{N}$, *such that* $s = \mathtt{curr}(\mathtt{succ}^k(\mathcal{P}_C))$.

A search engine $(\mathtt{succ}, \mathtt{curr})$ *is* systematic *if, for any* CSP C, *and* $i, j \in \mathbb{N}$ *with* $i \neq j$, *either* $\mathtt{succ}^i(\mathcal{P}_C) \neq \mathtt{succ}^j(\mathcal{P}_C)$, *or both are empty.*

Completeness simply says that if there are any solutions to the problem, at least one of them will be found using the search engine. This is a weaker definition of completeness than many authors use (cf. [Pearl, 1984] and [Tsang, 1993]), but it is easy to see how an algorithm can be changed to find all solutions. Simply turn each solution found into a negative constraint, eliminating it from Γ_C, and start the search again.

The condition of systematicity states that the same state can never show up twice when a search engine is running. Since the number of possible states is finite, this also guarantees termination. This definition of systematicity differs from the standard definition, which is that the same path in the search tree is never examined twice. Our version allows the same paths to be explored repeatedly, but each time in a different global state. This new definition is better applicable to modern search engines, which may in fact search the same path more than once, e.g. itera-

tive deepening [Korf, 1985] and dynamic backtracking [Ginsberg, 1993].

2.3 Procedures

Let us now turn our attention to procedural reasoning. Earlier we introduced two simple procedures that apply to the sample scheduling problem. One is unit propagation, which returns assignments that are forced by constraints that have not been satisfied yet. The other is a simplified version of PERT, which returns a failure indication anytime there is a cycle in the schedule or the current partial schedule violates a time bound.

In general, given a partial assignment to a CSP, a procedural reasoning method will either indicate which variable assignments are to be added or that the current partial assignment is a dead end. Although this is a simple description, it is powerful enough to capture almost any function and algorithm applicable to a CSP.

Our framework for extension procedures is therefore defined as follows:

Definition 2.9 *Let C be a* CSP. *An* extension procedure *for C is a function* $\mathtt{ext} : \mathcal{P}_C \rightarrow \mathcal{P}_C \cup \{\bot\}$, *such that if* $\mathtt{ext}(p) \neq \bot$, *then* $p \subseteq \mathtt{ext}(p)$.

An extension procedure is correct *if for any* $p \in \mathcal{P}_C$ *and* $s \in \Gamma_C$ *such that p extends to s,* $\mathtt{ext}(p) \subseteq s$.

The correctness criterion is based on the idea that procedures will only return direct consequences, seperating their role clearly from the role of the search engine. It simply states that if p can be extended to a specific solution, $\mathtt{ext}(p)$ can be extended to the same solution. This means that if $\mathtt{ext}(p) = \bot$, p cannot be extended to a solution, and if $\mathtt{ext}(p) = p'$, the only way to extend p to a solution is through p'.

If we look at the two sample procedures, we find that both are correct extension procedures. Assume the unit propagation procedure indicates that the partial assignment p should be extended to a larger assignment p'. If p could be extended to some solution s, the constraints that force the assignments in $p' - p$ must be satisfied in s, so we have $p \cup p' \subseteq s$.

To show that the PERT algorithm is correct, let us assume that it returns failure on some partial assignment p, and then show that p cannot be extended to a solution. If the failure was due to p having a cycle in the ordering, it is obvious that p cannot be extended to a solution, since no additional assignments can remove the cycle. If the failure is due to time limits, let us note that the PERT procedure calculates the earliest possible starting time for each job. For any partial ordering, additional assignments can only cause the earliest starting time for a job to move to a later time, never to an earlier time. Therefore, if the partial assignment p causes a given job to have its earliest starting time outside the time limits, p is a true dead end.

At this time, it should be pointed out that this correctness criterion is fairly strong in the sense that there are widely used procedureal methods that do not satisfy it. Examples of such procedural reasoning mechanisms are symmetry breaking and the pure literal rule. Our framework can easily be extended to include such algorithms, and there are corresponding theoretical results for the extended framework. Including those results is unfortunately beyond the scope of this paper, but they can be found elsewhere [Jónsson, 1996].

2.4 Combining Engines and Procedures

Having formalized the concepts of search engines and extension procedures, let us turn to the task of combining the two. The main problem is that procedures can potentially return information that does not fit within the data structures used by the search engine, causing the search engine to fail. To see how this may occur, let us look at a simple implementation of breadth-first search. The search engine uses a binary vector to keep track of which nodes at the current level have been explored. When it expands a new level, it generates a new binary vector. If a procedure returns a partial assignment that is not a member of the current level, there is no corresponding bit available to the search engine. The search engine must be allowed to disregard the result of the procedure, and just mark the original node as having been examined. In general, this problem is solved by allowing the search engine to choose a subset of the information returned by the extension procedure.

For the general case, let us thus define:

$$\mathcal{L}_C = \{S \in \mathcal{S}_C \mid \mathtt{succ}(S) \text{ is defined}\}$$

the set of all legal states for the search engine $(\mathtt{succ}, \mathtt{curr})$ working on problem C. The basic idea is that as long as the search engine does not encounter a state $S \notin \mathcal{L}_C$, it will not fail while solving C (although it may return the failure indication $\bot$). So if **ext** is an extension procedure, all we need to do is to require the result of invoking **ext** to be within this set of legal states.

Given a partial assignment p, a procedure **ext** will either return another partial assignment $p' = \mathtt{ext}(p)$,

or the failure indication $\perp$. Assuming the procedure is correct, this allows us to safely eliminate some of the partial assignments from the state of the search engine. If the procedure returns an extended partial assignment p', any alternative to that particular assignment is a dead end and can be removed. If the procedure returns $\perp$, the input assignment is a dead end, so it and all assignments that include it can safely be eliminated. Figure 2 shows exactly which nodes can be removed in each case.

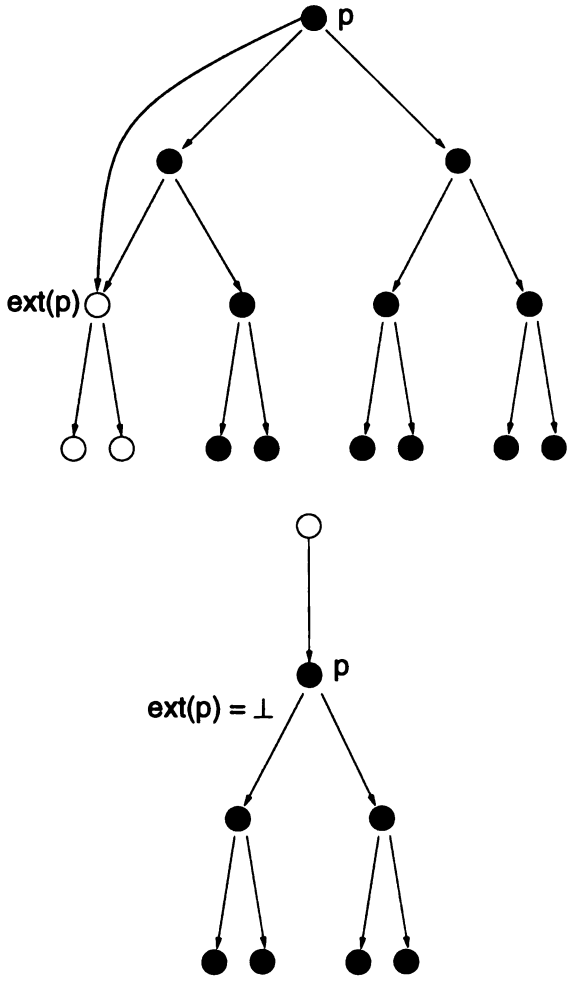


Figure 2: The set of nodes (colored gray) that can be eliminated from the search space when extension procedure ext is invoked on a partial assignment p.

Now that we know which nodes can be eliminated, describing how a search engine uses a procedure is easy. Let us look at a search engine $(\mathtt{succ}, \mathtt{curr})$ that is in state S, and a correct extension procedure ext. We apply the extension procedure to the current partial assignment $p = \mathtt{curr}(S)$, and it returns either another partial assignment p' or $\perp$. In each case, we know which nodes can safely be removed, so let us denote this set as E. The search engine must then update its state. It could update it to $\mathtt{succ}(S)$ as it would without the procedure, or it could update it to $\mathtt{succ}(S) - E$, if it is capable of representing that state. These are the two extremes; the search engine can in fact choose any set between those two, as long as it is in $\mathcal{L}_C$.

To make these definitions formal, let us first define the set of all descendents of a given partial assignment:

Definition 2.10 *Given a* CSP C *and a partial assignment* $p \in \mathcal{P}_C$, *the* set of descendents of p, *is:*

$$\delta(p) = \{p' \in \mathcal{P}_C \mid p' \supseteq p\}$$

For a correct extension procedure, we can now define the set of partial assignments that can safely be removed from the state of a search engine, since they don't lead to solutions:

Definition 2.11 *Given a partial assignment* p *and a correct extension procedure* ext, *the set* $\mathtt{elim}(\mathtt{ext}, p)$, *the set of partial assignments that can be eliminated from the search space, is:*

$$\mathtt{elim}(\mathtt{ext}, p) = \begin{cases} \delta(p) - \delta(\mathtt{ext}(p)); & \text{if } \mathtt{ext}(p) \neq \perp \\ \delta(p); & \text{if } \mathtt{ext}(p) = \perp \end{cases}$$

Finally, we can formally define how a search engine uses a procedure:

Definition 2.12 *Given a search engine* $(\mathtt{succ}, \mathtt{curr})$, *a* CSP C *and an extension procedure* ext *for* C, *we define a* search engine using procedure ext *as any search engine of the form* $(\mathtt{succ}_{\mathtt{ext}}, \mathtt{curr})$, *where* curr *is unchanged, and the function* $\mathtt{succ}_{\mathtt{ext}} : \mathcal{S}_C \to \mathcal{S}_C$ *is such that for any* $S \in \mathcal{L}_C$, *we have* $\mathtt{succ}_{\mathtt{ext}}(S) \in \mathcal{L}_C$, *and:*

$$\mathtt{succ}(S) - \mathtt{elim}(\mathtt{ext}, \mathtt{curr}(S)) \subseteq \mathtt{succ}_{\mathtt{ext}}(S) \subseteq \mathtt{succ}(S)$$

This definition is obviously valid, since there is always at least one legal state that can be chosen, namely $\mathtt{succ}(S)$. It should also be noted that the definition of a search engine using a procedure allows for certain flexibility in the choice of successor states, but in most cases the search engine will choose a minimal set for $\mathtt{succ}_{\mathtt{ext}}(S)$.

3 Theory

Having defined search engines, procedures and their interface, we now turn our attention to what properties a search engine will have when used with a procedure. The reasons for doing this are clear. The framework

allows search engines and procedures that are implemented separately to be used together. The possibility of conflict is real; experiments have shown that sophisticated search engines may lose systematicity if care is not taken. Theoretical conditions that will prevent this possibility of conflict are essential.

Given that procedures simply put us in a different part of the search space, the most obvious solution is to require the search engine to be globally complete and systematic, i.e. to be complete and systematic regardless of its starting point. The difference between the global properties and the original versions can best be explained by looking at an example of a search engine that is systematic but not globally systematic.

Imagine a search engine that systematically examines all possible partial assignments, keeping track of its progress by updating its state. It starts at the zero-information state $\mathcal{P}_C$, and then progresses through $\mathbf{succ}(\mathcal{P}_C)$, $\mathbf{succ}^2(\mathcal{P}_C)$, etc. Let us now imagine that for any state $S \notin \{\mathbf{succ}^k(\mathcal{P}_C) \mid k \in \mathbb{N}\}$, we have $\mathbf{succ}(S) = S$. If an extension mechanism produces a state outside the set of "regular" states it will get into an infinite loop. This search engine is systematic but not globally systematic.

The global concepts are captured in the following definition:

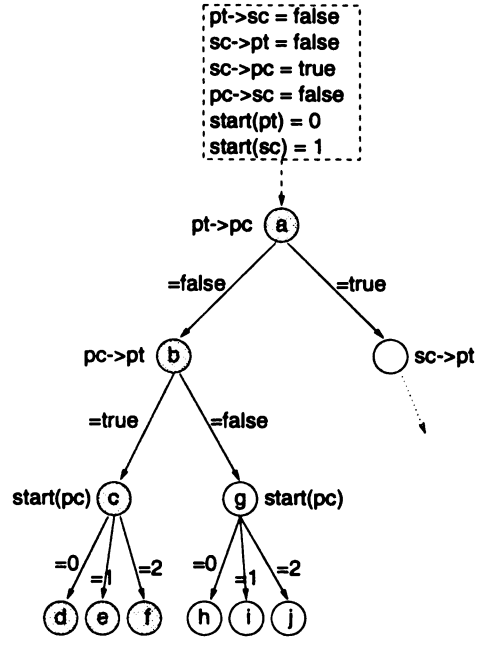
Definition 3.1 *A search engine* $(\mathbf{succ}, \mathbf{curr})$ *is* globally complete *if for any* CSP C, *and any* $S \in \mathcal{L}_C$, *such that* $S \cap \Gamma_C \neq \emptyset$, *there exists* $s \in S \cap \Gamma_C$ *and* $k \in \mathbb{N}$ *such that* $\mathbf{curr}(\mathbf{succ}^k(S)) = s$.

A search engine $(\mathbf{succ}, \mathbf{curr})$ *is* globally systematic *if for any* CSP C, *any* $S \in \mathcal{L}_C$, *and* $i, j \in \mathbb{N}$ *with* $i \neq j$, *either* $\mathbf{succ}^i(S) \neq \mathbf{succ}^j(S)$ *or both are empty.*

These are useful concepts that we will use later, but they are not sufficient to guarantee completeness and systematicity when an extension procedure is used. To see why, let us examine a specific version of iterative best first search. This search engine selects a set of candidates, and examines these in depth-first fashion. If that fails to give a solution, the set of candidates is expanded and the search is repeated.

To see how this is represented in a search engine, we define curr just as in depth first search, namely that $\mathbf{curr}(S)$ is the leftmost, deepest node in S that has no parents in S. The function succ is defined as follows:

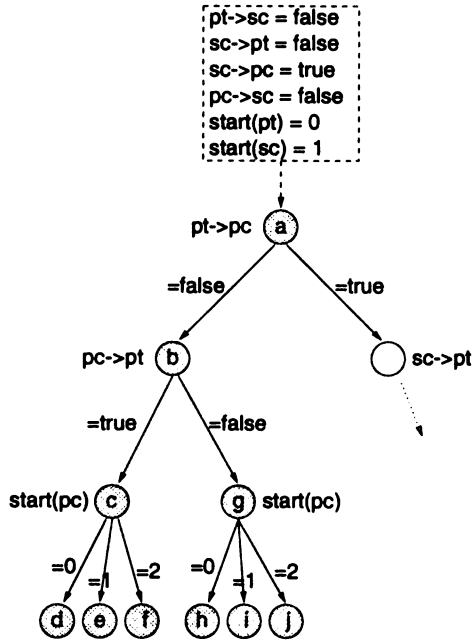
- $\mathbf{succ}(\mathcal{P})$ is the smallest subtree that spans the root node and all siblings of the leftmost leaf node.
- If S contains only the rightmost leaf node in the whole tree, let $\mathbf{succ}(S) = \emptyset$, since this leaf node is the final node to explore.
- If only one node is unexplored in the current iteration, i.e., S contains only one node p, other than the final node, let $\mathbf{succ}(S)$ be the smallest subtree spanning the root node, the single node p, all nodes left of p, and all siblings of the next node to the right.
- Otherwise, let $\mathbf{succ}(S) = \mathbf{succ}(S) - \{\mathbf{curr}(S)\}$.



n	$S = \mathbf{succ}^n(\mathcal{P})$	$\mathbf{curr}(S)$
i	$\{a, b, c, d, e, f\}$	a
$i+1$	$\{b, c, d, e, f\}$	b
$i+2$	$\{c, d, e, f\}$	c
$i+3$	$\{d, e, f\}$	d
..	etc.	..

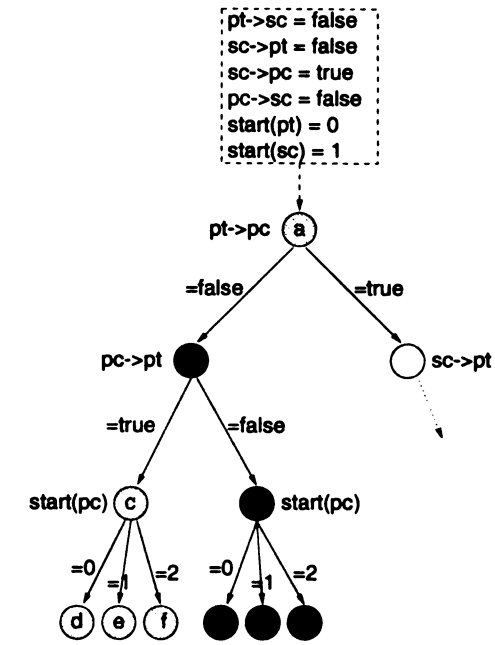
Figure 3: Portion of the first iteration of the iterative best first search engine, on the small scheduling problem. The assignments at the top of the figure are the decisions made higher up in the search tree. The gray nodes are the candidates examined in the current iteration, and the tables show how the search progresses during that iteration.

The search engine starts by exploring the leftmost subtree (gray in Figure 3) using depth first search. When this subtree has been explored, condition 3 in the definition of succ tells us to start with a new subtree, expanding the previous subtree to include a new set of leaf nodes. This new set of candidates is shown in



n	$S = \mathtt{succ}^n(\mathcal{P})$	$\mathtt{curr}(S)$
j	$\{a, b, c, d, e, f, g, h, i, j\}$	a
$j+1$	$\{b, c, d, e, f, g, h, i, j\}$	b
$j+2$	$\{c, d, e, f, g, h, i, j\}$	c
$j+3$	$\{d, e, f, g, h, i, j\}$	d
..	etc.	..

Figure 4: The second iteration in our iterative best first search engine, without a unit propagating procedure.



n	$S = \mathtt{succ}^n_{\mathtt{up}}(\mathcal{P}))$	$\mathtt{curr}(S)$
j	$\{a, b, c, d, e, f, g, h, i, j\}$	a
$j+1$	$\{b, c, d, e, f, g, h, i, j\}$	b
$j+2$	$\{c, d, e, f\}$	c
$j+3$	$\{d, e, f\}$	d
..	etc.	..

Figure 5: The second iteration in our iterative best first search engine, when a unit propagating procedure is used. The dark gray nodes denote the partial assignments eliminated by the procedure.

gray, in Figure 4. These iterations are then repeated until the search engine is halted by condition 2.

The search engine is easily seen to be globally complete. Given any set of candidates S, the engine will simply do depth first search on that subset. It is also easy to verify that the engine is globally systematic. Given a set S, this set will be examined in depth first fashion, and then a new iteration starts with the smallest set of iteration candidates that includes S and a new set of leaf nodes.

Let us now add unit propagation as an extension procedure up. The first iteration of the new search engine $\mathtt{succ}_{\mathtt{up}}$ is exactly the same as before, shown in Figure 3. In the second iteration, the search engine gets to the point where `pt->pc` has been assigned the value `false`, and the unit propagation procedure can conclude that `pc->pt` must be set to `true`, thus eliminating the other branch. If we look at Figure 5, we see exactly what happens. The dark colored nodes $\{b, g, h, i, j\}$ are the alternatives the procedure just eliminated, and those nodes can therefore be removed from the current state. But this puts the search engine into the same state as it was in the first iteration, causing an infinite loop. Needless to say, the search engine with the procedure $(\mathtt{succ}_{\mathtt{up}}, \mathtt{curr})$ is neither complete nor systematic.

It is worth pointing out that the culprit is not the idea of iterative best first search, but how the idea was implemented. The lesson to be learned is that care must be taken when search engines are implemented, and that even seemingly reasonable implementations can fail when used with a procedure.

The problem with this implementation of iterative depth-first search turns out be that reducing the set of possible candidates can actually set us back. This can be avoided by requiring the search engine to satisfy $\mathtt{succ}^k(S) \not\supseteq S$, for all $S \in \mathcal{L}_C$ and $k \in \mathbf{N}$. This is quite reasonable, as most systematic search engines make progress by gradually (but not necessarily monotonically) reducing the set of partial assignments that

may be looked at. Unfortunately this is also too weak. Figure 6 shows the problem that can be encountered.

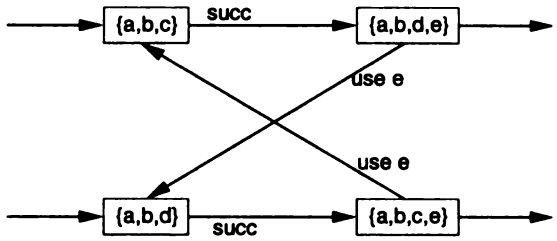


Figure 6: How a procedure may cause a loop in the search engine, even when $\mathtt{succ}(S)$ is not a superset of S.

Our goal is still to find a criterion that guarantees that a search engine works correctly with procedures. If we examine the two counterexamples found above, we note that in both cases it is actually systematicity that breaks. The search engines in our counterexamples do not eliminate all the valid solutions, they simply get stuck before they find one.

To find the appropriate condition to guarantee systematicity when procedures are used, we need to understand why a search engine is systematic. If we look at the simpler search engines, like depth-first search, it is easy to show that they are systematic But the more complex search engines are proved systematic using fairly involved theoretical proofs. When these proofs are examined we find that the search engines are actually systematic for different reasons. Each systematic search engine is making progress in some fashion, but the criterion for progress is different. To utilize this observation let us note that we can use an enumeration of the states to indicate what progress is, by making a lower number mean that progress has bee made. We then see that a search engine is systematic if updating each state counts as progress:

Lemma 3.2 *A search engine* $(\mathtt{succ}, \mathtt{curr})$ *is globally systematic if and only if, for any* CSP C*, there exists a function* $\mathtt{enum} : \mathcal{L}_C \to \mathbf{N}$*, such that* $\mathtt{enum}(\mathtt{succ}(S)) < \mathtt{enum}(S)$ *for all nonempty* $S \in \mathcal{L}_C$.

This observation gives us the criterion we have been looking for. For a search engine to remain systematic when working with a procedure, the results of using a procedure must be counted as progress. Since a procedure works by eliminating some of the partial assignments in the state, we simply require that changing a state to a smaller subset of that state must count as progress. More formally, if we are measuring progress using a function $\mathtt{enum} : \mathcal{L}_C \to \mathbf{N}$ such that for any states S and S' with $S' \subset S$ we have $\mathtt{enum}(S') < \mathtt{enum}(S)$, using a procedure will not prevent the search engine from continuing to make progress.

And when we have systematicity, it is easy to guarantee completeness: If a state contains solutions, either the current partial assignment must be a solution, or at least one solution remains in the updated state. This is enough, since a correct procedure will not eliminate any solutions.

We can now state our main theorem formally:

Theorem 3.3 *Let* $(\mathtt{succ}, \mathtt{curr})$ *be a search engine such that for any* CSP C*:*

1. *There exists a function* $\mathtt{enum} : \mathcal{L}_C \to \mathbf{N}$*, such that for any* $S, S' \in \mathcal{L}_C$*, with* $S' \subseteq \mathtt{succ}(S)$ *we have* $\mathtt{enum}(S') < \mathtt{enum}(S)$*, or* S' *and* S *are both empty.*

2. *For any* $S \in \mathcal{L}_C$ *such that* $S \cap \Gamma_C \neq \emptyset$*, either* $\mathtt{curr}(S) \in \Gamma_C$ *or* $\mathtt{succ}(S) \cap \Gamma_C \neq \emptyset$.

Then, for any correct extension procedure ext*, the search engine using* ext*, i.e.* $(\mathtt{succ}_{\mathtt{ext}}, \mathtt{curr})$*, satisfies the same conditions.*

The conditions in this theorem guarantee global systematicity and global completeness:

Lemma 3.4 *If a search engine* $(\mathtt{succ}, \mathtt{curr})$ *satisfies condition 1 in Theorem 3.3, it is globally systematic. If a search engine satisfies conditions 1 and 2 in Theorem 3.3, it is globally complete.*

It is worth noting that the converse does not hold, the conditions stated in the theorem are strictly stronger than global systematicity and global completeness. This was clearly demonstrated by the best-first search counterexample that was described earlier.

Using this last lemma and the main theorem, we get exactly what we wanted, namely conditions that guarantee completeness and systematicity for a search engine using a correct procedure.

Corollary 3.5 *If* $(\mathtt{succ}, \mathtt{curr})$ *satisfies the conditions in Theorem 3.3, and* ext *is a correct procedure, then* $(\mathtt{succ}_{\mathtt{ext}}, \mathtt{curr})$ *is complete and systematic.* ■

The conditions of Theorem 3.3 are fairly weak, in the sense that almost any known systematic search engine can be made to satisfy those conditions. In fact, most systematic search engines are covered by the following corollary:

Corollary 3.6 *Let* (succ, curr) *be a search engine that is monotonic in the sense that for any* CSP C *we have* $\mathbf{succ}(S) \subset S$ *for all* $S \in \mathcal{L}_C$. *If* (succ, curr) *satisfies condition 2 in Theorem 3.3, then* $(\mathbf{succ}_{\mathbf{ext}}, \mathbf{curr})$ *is systematic and complete, for any correct extension procedure* ext.

Finally, let us note that the conditions in Theorem 3.3 are not affected by the addition of a correct extension procedure. Therefore, we can continue adding procedures, as many as we want, and the search engine will still remain complete and systematic. This result allows us to design and implement general purpose search engines that can use libraries of seperately implemented extension procedures to speed up the search process. Such systems will have all the advantages of general and well-behaved reasoning engines, while retaining the benefits of using fast, special-purpose reasoning procedures.

4 Concluding Remarks

We have presented a theoretically sound framework that allows search engines to use extension procedures to speed up the search. We have also presented conditions are sufficient to guarantee correctness, completeness and systematicity, regardless of the number of correct procedures the search engine uses.

The reason our framework provides these results is that we have found the appropriate definitions for each entity in the framework. As we have already pointed out, certain concepts (e.g. abstract search engine) needed to be invented, while others (e.g. systematicity and completeness) were modified to be more generally applicable. Our framework will therefore provide a foundation on which further theoretical analysis of search engines can be built.

Finally, our framework gives us the tools we need to implement efficient, general CSP solvers that can use a library of extension procedures to speed up the search. As long as the solvers satisfy the conditions expressed in Theorem 3.3, and each of the procedures in the library is correct, the system will remain complete and systematic. This is invaluable in the search for better search engines and more powerful reasoning methods, and at the same time moves us closer to developing an efficient, general problem solver.

A Proofs

Lemma 3.2 *A search engine* (succ, curr) *is globally systematic if and only if, for any* CSP C, *there exists a function* $\mathbf{enum} : \mathcal{L}_C \to \mathbf{N}$, *such that* $\mathbf{enum}(\mathbf{succ}(S)) < \mathbf{enum}(S)$ *for all nonempty* $S \in \mathcal{L}_C$.

Proof. For one direction, assume (succ, curr) is globally systematic. Then if we define a relation $<_{\mathbf{succ}}$ on $\mathcal{L}_C$ by $\mathbf{succ}(S) <_{\mathbf{succ}} S$, we find that $<_{\mathbf{succ}}$ is a partial order. Now let enum be an enumeration of some topological sort of this partial order. If $S \in \mathcal{L}_C$, we get $\mathbf{succ}(S) <_{\mathbf{succ}} S$, and thus $\mathbf{enum}(\mathbf{succ}(S)) < \mathbf{enum}(S)$.

Conversely, assume there is such an enum. Let $S \in \mathcal{L}_C - \{\emptyset\}$ and $i, j \in \mathbf{N}$ with $i \neq j$. Without loss of generality we can assume $i > j$. We can then write $\mathbf{succ}^i(S) = \mathbf{succ}^{i-j}(\mathbf{succ}^j(S))$. Applying the fact $\mathbf{enum}(\mathbf{succ}(S)) < \mathbf{enum}(S)$, $i - j$ times, we get $\mathbf{enum}(\mathbf{succ}^i(S)) < \mathbf{enum}(\mathbf{succ}^j(S))$, and thus $\mathbf{succ}^i(S) \neq \mathbf{succ}^j(S)$. ■

Theorem 3.3 *Let* (succ, curr) *be a search engine such that for any* CSP C:

1. *There exists a function* $\mathbf{enum} : \mathcal{L}_C \to \mathbf{N}$, *such that for any* $S, S' \in \mathcal{L}_C$, *with* $S' \subseteq \mathbf{succ}(S)$ *we have* $\mathbf{enum}(S') < \mathbf{enum}(S)$, *or* S' *and* S *are both empty.*

2. *For any* $S \in \mathcal{L}_C$ *if* $S \cap \Gamma_C \neq \emptyset$, *either* $\mathbf{curr}(S) \in \Gamma_C$ *or* $\mathbf{succ}(S) \cap \Gamma_C \neq \emptyset$.

Then, for any correct extension procedure ext, *the search engine using* ext, *i.e.* $(\mathbf{succ}_{\mathbf{ext}}, \mathbf{curr})$, *satisfies the same conditions.*

Proof. Let (succ, curr) be a search engine that satisfies these conditions, let C be a CSP and ext a correct extension procedure for it.

For the first condition we use the same enumeration enum. Let S be a nonempty member of $\mathcal{L}_C$, and note that $\mathbf{succ}_{\mathbf{ext}}(S) \subseteq \mathbf{succ}(S)$ by definition. If $S' \in \mathcal{L}_C$ with $S' \subseteq \mathbf{succ}_{\mathbf{ext}}(S)$, we get $S' \subseteq \mathbf{succ}(S)$, and thus $\mathbf{enum}(S') < \mathbf{enum}(S)$. If S is empty, any $S' \subseteq \mathbf{succ}_{\mathbf{ext}}(S)$ is also empty.

For the second condition, let $S \in \mathcal{L}_C$ such that $S \cap \Gamma_C \neq \emptyset$. If $\mathbf{curr}(S)$ is a solution we are done. Otherwise we know $\mathbf{succ}_{\mathbf{ext}}(S) \supseteq \mathbf{succ}(S) - [\delta(\mathbf{curr}(S)) - \delta(e(\mathbf{curr}(S)))]$. Since ext is correct, there are no solutions in $\delta(\mathbf{curr}(S)) - \delta(\mathbf{ext}(\mathbf{curr}(S)))$ so any $s \in \mathbf{succ}(S) \cap \Gamma_C$ is also in $\mathbf{succ}_{\mathbf{ext}}(S) \cap \Gamma_C$. ■

Lemma 3.4 *If* (succ, curr) *satisfies the conditions in Theorem 3.3, then* (succ, curr) *is complete and systematic.*

Proof. Condition 1 states that $\mathbf{enum}(S') < \mathbf{enum}(S)$ for any $S' \subseteq \mathbf{succ}(S)$, in particular for $S' = \mathbf{succ}(S)$, so we get systematicity immediately from Lemma 3.2.

For completeness, let us assume that $(\mathbf{succ}, \mathbf{curr})$ is not complete, i.e., Γ_C is not empty, and $\mathbf{curr}(\mathbf{succ}^n(\mathcal{P})) \notin \Gamma_C$ for all $n \in \mathbf{N}$. Using condition 2, we can then show that $\mathbf{succ}^n(\mathcal{P}) \cap \Gamma_C \neq \emptyset$ for all $n \in \mathbf{N}$. But we just showed that $(\mathbf{succ}, \mathbf{curr})$ is systematic, and since there are only finitely many members in $\mathcal{S}_C$, we have a contradiction. Thus $(\mathbf{succ}, \mathbf{curr})$ must be complete. ■

Corollary 3.6 *Let* $(\mathbf{succ}, \mathbf{curr})$ *be a search engine that is monotonic in the sense that for any* CSP C *we have* $\mathbf{succ}(S) \subset S$ *for all* $S \in \mathcal{L}_C$. *If* $(\mathbf{succ}, \mathbf{curr})$ *satisfies condition 2 in Theorem 3.3, then* $(\mathbf{succ}_{\mathbf{ext}}, \mathbf{curr})$ *is systematic and complete, for any correct extension procedure* **ext**.

Proof. $(\mathbf{succ}, \mathbf{curr})$ satisfies both conditions in Theorem 3.3. ■

Acknowledgements

The authors would like to thank all the members of the Computational Intelligence Research Laboratory for their suggestions and advice regarding this work.

This work has been supported by the Air Force Office of Scientific Research under contract F49620-92-J-0384 and by ARPA/Rome Labs under contracts F30602-91-C-0036, F30602-93-C-00031 and F30602-95-I-0023.

References

[Crawford and Baker, 1994] James M. Crawford and Andrew B. Baker. Experimental results on the application of satisfiability algorithms to scheduling problems. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 1994.

[Crawford *et al.*, 1996] James M. Crawford, Matthew L. Ginsberg, Eugene M. Luks, and Amithaba Roy. Symmetry breaking predicates for search problems. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR '96)*, 1996.

[Ginsberg, 1993] Matthew L. Ginsberg. Dynamic backtracking. *Journal of Artificial Intelligence Research*, 1:25–46, 1993.

[Green, 1969] C. C. Green. *The application of theorem proving to question answering systems.* PhD thesis, Stanford University, Stanford, CA, 1969.

[Hillier and Lieberman, 1986] Frederick S. Hillier and Gerald J. Lieberman. *Introduction to Operations Research.* Holden-Day, Oakland,CA, 1986.

[Jónsson, 1996] Ari K. Jónsson. *Procedural Reasoning in Constraint Satisfaction.* PhD thesis, Stanford University, Stanford, CA, 1996. (To appear.)

[Korf, 1985] Richard E. Korf. Depth-first iterative deepening: An optimal admissible tree search. *Artificial Intelligence*, 27:97–109, 1985.

[Langley, 1992] Pat Langley. Systematic and nonsystematic search strategies. In *Artificial Intelligence Planning Systems: Proceedings of the First International Conference*, pages 145–52. Morgan Kaufmann, 1992.

[Maluszynski *et al.*, 1993] J. Maluszynski, S. Bonnier, J. Boye, F. Kluzniak, A. Kagedal, and U. Nilsson. Logic programs with external procedures. In K. R. Apt, J.W. de Bakker, and J. J. M. M. Rutten, editors, *Logic Programming Constraints, Functions, and Objects*, pages 21–48. MIT Press, Cambridge, MA, 1993.

[Myers, 1994] Karen L. Myers. Hybrid reasoning using universal attachment. *Artificial Intelligence*, 67:329–375, 1994.

[Pearl, 1984] Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley, Reading, MA, 1984.

[Selman *et al.*, 1993] Bart Selman, Henry A. Kautz, and Bram Cohen. Local search strategies for satisfiability testing. In *Proceedings 1993 DIMACS Workshop on Maximum Clique, Graph Coloring, and Satisfiability*, 1993.

[Smith and Cheng, 1993] Stephen F. Smith and Cheng-Chung Cheng. Slack-based heuristics for constraint satisfaction scheduling. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 139–44, 1993.

[Stallman and Sussman, 1977] Richard M. Stallman and Gerald J. Sussman. Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. *Artificial Intelligence*, 9:135–96, 1977.

[Tsang, 1993] E. Tsang. *Foundations of Constraint Satisfaction.* Academic Press, London, 1993.

Robotics

Representing Sensing Actions: The Middle Ground Revisited

Keith Golden **Daniel Weld**
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195
{kgolden, weld}@cs.washington.edu

Abstract

To build effective planning systems, it is crucial to find the right level of representation: too impoverished, and important actions and goals are impossible to express; too expressive, and planning becomes intractable. Within the classical framework, Pednault's ADL (Pednault 1986) provided a happy compromise between the impoverished STRIPS representation and the expensive situation calculus.

Among languages handling sensing actions and information goals, there is a similar spectrum of expressiveness. UWL, an extension of STRIPS, can't express goals like "Rename the file `paper.tex` to `kr.tex`." Nor can it represent universally quantified goals or effects. At the other extreme are elegant languages (Morgenstern 1987; Moore 1985; Levesque 1996) for which effective planners don't exist.

In this paper, we combine elements of UWL and ADL, to define SADL: a middle-ground representation for sensing actions. Underlying our language are two insights, missing from UWL: 1) Knowledge goals are inherently temporal. 2) Knowledge preconditions are unnecessary for an important class of domains (those obeying a Markov property). SADL is expressive enough to encode the rich domain theory of the Internet Softbot, including hundreds of UNIX and Internet operators; yet it supports tractable inference by planners such as XII (Golden, Etzioni, & Weld 1994; 1996).

1 INTRODUCTION

One of the stumbling blocks to past research in planning with incomplete information has been inadequate or imprecisely defined languages for representing information goals and sensing actions. Many researchers have devised formalisms for reasoning about knowledge and action (Moore 1985; Morgenstern 1987; 1988; Drummond 1989; Davis 1993; Scherl & Levesque 1993; Levesque 1996), but those languages are too expressive to be used in practical planning algorithms. UWL (Etzioni *et al.* 1992) offered a more tractable representation (based on STRIPS) that was tailored to current planning technology, but as Levesque (Levesque 1996) observes, the semantics of UWL are unclear — the definitions were made relative to a specific planning algorithm. In our efforts to define a semantics for UWL, we determined that UWL confused information goals with maintenance goals, and conflated knowledge goals with knowledge preconditions. Furthermore, years of experience with UWL convinced us that it wasn't expressive enough to fully handle the real-world domains (*e.g.*, UNIX and the Internet) for which it was intended. Since UWL didn't support universal quantification or conditional effects, it could not correctly represent the UNIX command `ls`, which lists all files in a directory, or `rm *`, which deletes all writable files.

Information	Expressiveness ⟶		
Complete	STRIPS	ADL	Situation Calculus
Incomplete	UWL	**SADL**	Moore *et al*

In this paper, we define a new action representation language, SADL,[1] that combines ideas from UWL with those from Pednault's ADL (Pednault 1989; 1986). Just as ADL marked the "middle ground" on the tractability spectrum between STRIPS and the situation calculus, SADL offers an advantageous combination of expressiveness and efficiency. Since SADL supports universally quantified information goals and universally quantified, conditional, observational effects, it is expressive enough to represent hundreds of UNIX and Internet commands. Indeed, four years of painful experience writing and debugging the Internet Softbot (Etzioni & Weld 1994) knowledge base forced us

[1] SADL (pronounced "Saddle") stands for "Sensory Action Description Language."

to uncover and remedy some subtle confusions about information goals:

- In a dynamic world, knowledge goals are inherently *temporal* — If proposition P is true at one time point and false in another, which time point do we mean when we ask about P's truth value? Since UWL has limited provision to make temporal distinctions, it cannot encode an important class of goals. In particular, UWL cannot express goals that require causal change to attributes used to designate objects, *e.g.*"Rename the file `paper.tex` to `kr.tex`." (See Sections 2.2 and 2.3 for the SADL solution)
- We identify a large class of domains, called *Markov domains*, and argue that actions in these domains are best encoded *without knowledge preconditions*. The multiagent scenarios that inspired Moore, Morgenstern, and others are not Markov, but UNIX and much of the Internet are. While SADL discourages knowledge *preconditions* it recognizes the need for knowledge *subgoals*. (Section 2.4 elaborates).

1.1 ROADMAP

Section 2 describes problems with the UWL formulation of knowledge goals and presents the SADL solution. In Section 3 we discuss observational effects of actions, and causal effects, which can decrease the agent's knowledge about the world. We also demonstrate the representational adequacy of SADL by presenting an encoding of the UNIX `ls -a` command. In Section 4 we discuss temporal projection in SADL. In Section 5 we demonstrate that the SADL formalism is expressive enough to represent many interesting actions. Section 6 argues that SADL's expressive power comes at a reasonable price — reasoning is tractable. We conclude with a discussion of related work in Section 7 and a summary in Section 8.

2 KNOWLEDGE GOALS AND PRECONDITIONS

In UWL, preconditions and goals were limited to conjunctions of literals, each annotated with one of three tags: **satisfy**, **hands-off**, and **find-out**. The SADL action language is based on UWL, but uses a different set of annotations: **satisfy**, **hands-off**, and **initially**, which provide a cleaner semantics for information goals and greater expressive power; additionally, SADL uses unannotated literals to designate preconditions that don't depend on the agent's knowledge. Furthermore, SADL supports universal quantification and conditional effects, both of which have interesting ramifications in the context of incomplete information. We proceed by reviewing UWL, uncovering some confusions, presenting the SADL solution, and sketching the formal semantics.

In UWL (and in SADL) individual literals have truth values expressed in a three-valued logic: `T`, `F`, `U` (unknown). Free variables are implicitly existentially quantified, and the quantifier takes the widest possible scope.[2] For example, **satisfy**(in.dir (f, `tex`), `T`)[3] means "Ensure that there's at least one file in directory `tex`." Truth values can also be represented by variables. For example, **satisfy**(in.dir (`myfile`, `tex`), tv) means "Find out whether or not `myfile` is in `tex`."

Although the semantics of UWL was defined procedurally (Etzioni *et al.* 1992), we provide SADL's semantics in terms of the situation calculus. The situation calculus (McCarthy & Hayes 1969) is a first-order logic used to capture changes to the world that come about by the execution of actions. A *fluent* is a proposition whose truth value changes over time. Every fluent, $\varphi(x)$, takes an additional argument, namely a situation, s. $\varphi(x, s)$ represents the statement that $\varphi(x)$ holds in situation s. By convention, s is always the last argument of φ, so we will freely add or drop the s, depending on whether we are referring to φ in a particular situation. Thus, if $\text{in.dir}(f, d)$ means file f is in directory d, $\text{in.dir}(f, d, s)$ means this fact holds in situation s. All state changes are assumed to result from the execution of actions. The special function DO is used to describe these changes: $\text{DO}(a, s)$ returns the situation resulting from executing action a in situation s. We use $\{a\}_1^n$ to represent the sequence of actions $a_1; a_2; \ldots; a_n$. $\text{DO}(\{a\}_1^n, s)$ denotes nested application $\text{DO}(a_n, \text{DO}(a_{n-1}, \ldots, \text{DO}(a_1, s)))$, *i.e.*, the result of executing the entire sequence, starting in situation s. We use s_n as a shorthand for $\text{DO}(\{a\}_1^n, s_0)$.

Our formulation of SADL is based on Scherl and Levesque's (Scherl & Levesque 1993) solution to the frame problem for knowledge-producing actions. We adopt their completeness assumptions, and their formulation of incomplete knowledge, and thus their results (i.e. the persistence of knowledge and of ignorance) hold for us as well. Incomplete knowledge is defined in terms of the standard possible-worlds semantics, where $\text{K}(s', s)$ means that if the situation is s, then it is consistent with the agent's knowledge to believe that the situation could in fact be s'. In other words, $\{s' | \text{K}(s', s)\}$ denotes the set of all possible worlds consistent with the agent's knowledge in situation s. We assume that an agent's knowledge is correct, so the actual situation is always considered possible by the agent ($\forall s.\text{K}(s, s)$), and we assume that situations only change when the agent executes an ac-

[2] Explicit quantifiers can be used to indicate a narrower scope.

[3] For notational convenience, an omitted truth value defaults to `T`, so this could be rewritten as **satisfy**(in.dir (f, `tex`)). We use this shorthand in the remainder of the paper. Italicized lower-case symbols, such as f, denote variables. Symbols in `typewriter` font denote constants. Annotations are in **bold**.

tion. We define $\text{KNOW}(\varphi, s) \stackrel{\text{def}}{=} \forall s'.\text{K}(s', s) \Rightarrow \varphi(s')$, *i.e.*, φ is true in all worlds consistent with the agent's knowledge.

As we mentioned, SADL uses a three-valued logic (T, F, U) to represent knowledge. The relation between these truth values and KNOW is straightforward. If φ has the truth value T, then $\text{KNOW}(\varphi)$. If φ has the truth value F, then $\text{KNOW}(\neg\varphi)$. If the truth value is U, then $\neg\text{KNOW}(\varphi) \wedge \neg\text{KNOW}(\neg\varphi)$.

2.1 SATISFACTION AND MAINTENANCE GOALS

The goal **satisfy**(P) indicates a traditional goal (as in ADL): achieve P by whatever means possible. In the presence of incomplete information, we make the further requirement that the agent knows that P is true. We define $\text{GOAL}(G, s_0, \{a\}_1^n)$ to mean that goal G is achieved in the situation resulting from executing plan $\{a\}_1^n$ in situation s_0; since we assume the agent's knowledge is correct, it is sufficient to state that the agent knows P:

$$\text{GOAL}(\textbf{satisfy}(P, \text{T}), s_0, \{a\}_1^n) \stackrel{\text{def}}{=} \text{KNOW}(P, s_n) \quad (1)$$

$$\text{GOAL}(\textbf{satisfy}(P, \text{F}), s_0, \{a\}_1^n) \stackrel{\text{def}}{=} \text{KNOW}(\neg P, s_n) \quad (2)$$

$$\text{GOAL}(\textbf{satisfy}(P, tv), s_0, \{a\}_1^n) \stackrel{\text{def}}{=} \begin{array}{l}\text{KNOW}(P, s_n) \vee \\ \text{KNOW}(\neg P, s_n)\end{array} \quad (3)$$

Note that when given an (existentially quantified) variable as truth value, a **satisfy** goal requires that the agent learn whether the proposition is true or false (which could be achieved by making it true or false). Equation 3 is a slight simplification; if several fluents in a goal use the *same* variable, tv, then they should all have the same truth value. The above definition fails to capture such correlations. We don't discuss correlated truth values in this paper, so for clarity, we omit these variable constraints in the remainder of the paper. However, we show them below for **satisfy** goals. Variable constraints in the other definitions follow the same form:

$$\text{GOAL}(\textbf{satisfy}(P, tv), s_0, \{a\}_1^n) \stackrel{\text{def}}{=}$$
$$\text{KNOW}(P \wedge tv = \text{T}, s_n) \vee \text{KNOW}(\neg P \wedge tv = \text{F}, s_n) \quad (4)$$

The **hands-off** annotation indicates a maintenance goal that prohibits the agent from changing the fluent in question.

$$\text{GOAL}(\textbf{hands-off}(P), s_0, \{a\}_1^n) \stackrel{\text{def}}{=}$$
$$\forall s \in \text{State-History}.[P(s) \Leftrightarrow P(s_0)] \quad (5)$$

By State-History we mean the set of $n+1$ situations produced during execution of $\text{DO}(\{a\}_1^n, s_0)$ (including both s_0 and s_n). Thus, the definition of **hands-off** requires that P not change value during execution of the plan. (Etzioni *et al.* 1992) noted that together, **satisfy**+**hands-off** can be used to indicate a "look but don't touch" goal: the agent may sense the fluent's value, but is forbidden to change it. While **hands-off** goals are clearly useful, we argue that they are an overly restrictive way of specifying *knowledge* goals. In particular, they outlaw changing the value of a fluent after it has been sensed.

2.2 KNOWLEDGE GOALS ARE INHERENTLY TEMPORAL

Before explaining the SADL approach to knowledge goals, we discuss the UWL **find-out** annotation. **find-out** is problematic because the original definition was in terms of a particular planning algorithm (Etzioni *et al.* 1992). The motivation for **find-out** was the existence of goals for which **hands-off** is too restrictive, but **satisfy** alone is too permissive. For example, given the goal "Tell me what files are in directory `tex`," executing `rm tex/*` and reporting "None" would clearly be inappropriate. But what about the conjunctive goal "Free up some disk space and tell me what files are in directory `tex`"? In this case *excluding* the `rm` seems inappropriate, since it may be necessary in service of freeing disk space. Yet the knowledge that the directory is now empty is relevant to the information goal. Proponents of **find-out** argued that `rm` was unacceptable for the first goal, but acceptable in service of the conjunction (Etzioni *et al.* 1992). We contend that this definition is unclear and unacceptable; a plan that satisfies the conjunction $A \wedge B$ should also be a solution to A.

While the examples used to justify the original **find-out** definition are evocative, their persuasive powers stem from *ambiguity*. At what *time point* do we wish to know the directory contents? Before freeing disk space, afterward, or in between? Since fluents are always changing, a general information goal requires two temporal arguments: the time a fluent is sensed, and the time the sensed value is to be reported. *E.g.*, one can ask "Who was president in 1883," or "Tell me tomorrow who was president today."

Since planning with an explicit temporal representation is slow, our quest for the "middle ground" along the expressiveness / tractability spectrum demands a minimal notion of time that captures most common goals. We limit consideration to two time points: the time when a goal is given to the agent, and the time the agent gives his reply. Note that **satisfy**(P, tv) (Equation 3) allows one to specify the goal of knowing P's truth value at this latter time point. To specify the goal of sensing a fluent at the time the goal is given, we introduce the annotation **initially**.

$$\text{GOAL}(\textbf{initially}(P, tv), s_0, \{a\}_1^n) \stackrel{\text{def}}{=}$$
$$[\forall s \in \text{ORIG}_n.\ P(s)] \vee [\forall s \in \text{ORIG}_n.\ \neg P(s)] \quad (6)$$

We use ORIG_n (Figure 1) to represent the agent's knowledge in s_n *about the past situation* s_0, *i.e.*, the set of situations indistinguishable

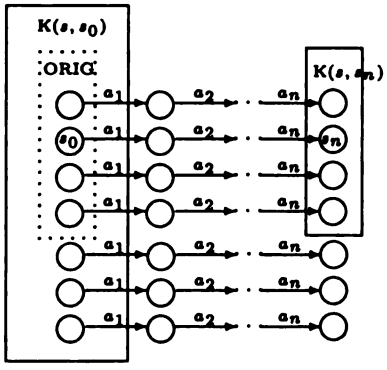


Figure 1: The region surrounded by dotted lines represents the set ORIG_n, the set of states indistinguishable from s_0, based on the agent's knowledge in state s_n. ORIG_n is a subset of $\{s \mid \text{K}(s, s_0)\}$, the states that were consistent with the agent's knowledge in s_0, since the agent has learned more about what originally held, but has not forgotten anything it knew originally.

from s_0 after execution of the plan: $\text{ORIG}_n = \{s \mid \text{K}(\text{DO}(\{a\}_1^n, s), \text{DO}(\{a\}_1^n, s_0))\}$. Thus the definition of **initially** states that when the agent has finished executing the plan, he will know whether P was true or false when he started. **initially**(P) is not achievable by an action that changes the fluent P, since such an action only obscures the initial value of P. However, changing P after determining its initial value is fine. By combining **initially** with **satisfy** we can express "tidiness" goals (modify P at will, but restore its initial value by plan's end) (Weld & Etzioni 1994). Furthermore, we can express goals such as "Find the the file currently named `paper.tex`, and rename it to `kr.tex`," which are impossible to express in UWL. Since UWL can't make temporal distinctions, there is no way to ask for the past value of a fluent without also requiring that the fluent have the same value when the reply is given, so any goal of the form "Find some x such that $P(x)$, and make $P(x)$ false" is inexpressible in UWL.

2.3 UNIVERSALLY QUANTIFIED GOALS

When defining universally quantified goals, one must again be specific with respect to time points: does the designator specifying the Herbrand universe refer to s_0 or s_n? Since SADL allows an arbitrary goal description to be used to scope a universally quantified goal, one can specify a wide range of requests. For example, suppose an agent is given the goal of seeing to it that all files in directory `tex` are compressed. What plans satisfy the goal? It depends on what the request really means. In SADL, one can write one of the following precise versions, thus eliminating the ambiguity.

1. Ensure that all files, which were initially in `tex`, end up being compressed: $\forall f$ **initially**(in.dir (f, `tex`)) ⇒ **satisfy**(compressed (f)). Executing **compress** `tex/*` solves this goal, as does executing **mv** `tex/* temp` then **compress** `temp/*`.
2. Ensure that all files, which end up in `tex`, end up being compressed: $\forall f$ **satisfy**(in.dir (f, `tex`)) ⇒ **satisfy**(compressed (f)). Executing **compress** `tex/*` solves this goal, but so does **rm** `tex/*`!
3. Determine if all files, initially in `tex`, were initially compressed: $\forall f$ **initially**(in.dir (f, `tex`)) ⇒ **initially**(compressed (f)).
4. Determine if all files, in `tex` at the end of execution, were initially compressed: $\forall f$ **satisfy**(in.dir (f, `tex`)) ⇒ **initially**(compressed (f)). This is equivalent to $\forall f$ **initially**(compressed (f), F) ⇒ **satisfy**(in.dir (f, `tex`), F), *i.e.* ensure that all files *not* initially compressed do *not* end up in `tex`.

The first example seems the most likely interpretation of the goal in this case, but it still leaves something to be desired, since the user may not want the files moved from `tex`. We can easily state the additional requirement that the files not be moved (**hands-off**(in.dir (f, `tex`))), or that they be returned to `tex` by the end (**satisfy**(in.dir (f, `tex`))). We should be careful not to make goals overly restrictive, though. If the desire is that the agent should fail if there's no way to compress the files without moving them, then adding such restrictions is correct. If the desire is merely that the agent should avoid moving the files unnecessarily, then we want the original solution, with some background preference to minimize unnecessary changes. Such background preferences could be expressed in terms of a utility function over world states (Raiffa 1968), a measure of plan quality (Pollack 1992; Wilkins 1988), or an explicit notion of harm (Weld & Etzioni 1994).

Note that even if we decide to forbid moving the files from `tex`, there are still other actions, such as deleting all the files in `important/papers`, or sending threatening email to `president@whitehouse.gov` that haven't been excluded. This is a general problem with satisficing plans: anything goes as long as the goal is achieved. Specifying all the undesired outcomes with every goal would be tedious and error-prone. A better solution is to separate the criteria of goal satisfaction from background preferences, as is done in (Williamson & Hanks 1994; Haddawy & Hanks 1993; Weld & Etzioni 1994).

Given the appropriate annotations on fluents, which provide temporal information, the semantics of ∀ goals is straightforward:

$$\text{GOAL}(\forall \vec{x}.P, s_0, \{a\}_1^n) \stackrel{\text{def}}{=} \forall \vec{x}.\text{GOAL}(P, s_0, \{a\}_1^n) \quad (7)$$

$$\text{GOAL}(P \Rightarrow Q, s_0, \{a\}_1^n) \stackrel{\text{def}}{=} \begin{array}{l}\text{GOAL}(P, s_0, \{a\}_1^n) \Rightarrow \\ \text{GOAL}(Q, s_0, \{a\}_1^n)\end{array} \quad (8)$$

Logical operators such as ∧, ∨, and ∃ follow the same form as above.

2.4 KNOWLEDGE PRECONDITIONS CONSIDERED HARMFUL

Moore (Moore 1985) identified two kinds of knowledge preconditions an agent must satisfy in order to execute an action in support of some proposition P: First, the agent must know a rigid designator (*i.e.*, an unambiguous, executable description) of the action. Second, the agent must know that executing the action will in fact achieve P. Subsequent work, *e.g.* (Morgenstern 1987), generalized this framework to handle scenarios where multiple agents reasoned about each other's knowledge.

In the interest of tractability, we take a much narrower view, assuming away Moore's first type of knowledge precondition and refuting the need for his second type. Our argument occupies the remainder of this section, but the summary is that there is a large class of domains, those obeying a ***Markov property***, for which actions are best encoded without knowledge preconditions. While the multiagent scenarios considered by Moore and Morgenstern are not Markov, UNIX and much of the Internet are.

We start the argument by assuming away Moore's first type of knowledge precondition. We define actions as programs that can be executed by a robot or softbot, without the need for further reasoning. In this view, ***all*** actions are rigid designators. `dial` (combination(`safe`)) is not an admissible action, but `dial(31-24-15)` is. Lifted action schemas, *e.g.* `dial(`x`)`, are not rigid designators, but it is easy to produce one by substituting a constant for x. Thus Moore's first type of knowledge precondition vanishes.

Moore's second type of knowledge precondition presupposes that an action in a plan must provably succeed in achieving a desired goal. This is a standard assumption in classical planning, but is overly restrictive given incomplete information about the world; enforcing this assumption by adding knowledge preconditions to actions is inappropriate. For example, if knowledge of the safe's combination is a precondition of the `dial` action, then it becomes impossible for a planner to solve the goal "find out whether the combination is 31-24-15" by dialing that number, since before executing the `dial` action, it will need to satisfy that action's precondition of finding out whether 31-24-15 is the right combination![4]

On the other hand, it is often necessary for an agent to plan to obtain information, such as the combination of a safe, either to reduce search or to avoid dangerous mistakes. These knowledge ***subgoals***, naturally, have a temporal component, but the only time point of interest is the moment the action is executed. For example, the goal of knowing the safe's combination could be satisfied by watching another agent open the safe, but it might also be satisfied by changing the combination to some known value (for instance, at some earlier time when the safe is open).

We say that an action is ***Markov*** if its effects depend only on the state of the world at the time of execution. Note that simple mechanical and software systems are naturally encoded as Markov, while multiagent systems are typically ***not***, because it is useful to endow one's model of another agent with state (*i.e.*, I know that Bill knew ...). If all actions in a domain are Markov, then all knowledge sub-goals will be of the same form: 1) The agent needs to know the value of some fluent at the time the action is to be executed, and 2) it doesn't matter if the agent affects the fluent while obtaining its value.[5] These requirements for knowledge sub-goals are met by the SADL definition of **satisfy** (Equation 3),[6] if we regard the action sequence $\{a\}_1^n$ as a plan to achieve the preconditions of action a_{n+1}.

The Markov assumption for actions yields a substantially simpler representation of change than those defined by Moore and Morgenstern. While their theories are more appropriate for complex, multi-agent domains, SADL gains tractability while retaining enough expressive power to model many important domains.

3 EFFECTS

Like UWL, SADL divides effects into those that change the world, annotated by **cause**, and those that merely report on the state of the world, annotated by **observe**. Because it lacked universal quantification, UWL couldn't even correctly model UNIX `ls`. SADL goes beyond UWL by allowing both observational and

[4]Note that eliminating the knowledge precondition from the `dial` action also allows the unhurried agent to devise a plan to enumerate the possible combinations until he finds one that works. Indeed, the Internet Softbot (Etzioni & Weld 1994) follows an analogous strategy when directed to find a particular user, file or a web page, whose location is unknown. If `finger` and `ls` included knowledge preconditions, then the actions would be useless for locating users and files.

[5]The reader may object that (nonrigid) indexical references could appear as preconditions to actions. For example, suppose that running Netscape requires that *the* file `netscape.bookmarks` be in a given directory. It is not sufficient that *a* file of that name be there, because renaming `paper.tex` to `netscape.bookmarks` would cause Netscape to fail. But this example makes it clear that the proposed preconditions of Netscape are simply underspecified. They should be "The directory contains a file named `netscape.bookmarks`, which is a valid bookmarks file, and ..." This is just the qualification problem (McCarthy 1980) in disguise. Granted, it will usually be impossible (or undesirable) to model all such preconditions.

[6]A justification that might be given for **initially** or **hands-off** preconditions is to minimize destructive actions used by an agent to satisfy a goal (*i.e.* don't use `mv` to find out the name of a file). We agree on the need for reasoning about *plan quality*, but an accurate theory of action should distinguish action preconditions from *user preferences*.

causal effects to have universal quantification and secondary preconditions.

3.1 OBSERVATIONAL EFFECTS

Executing actions with observational effects assigns values to *runtime* variables that appear in those effects. By using a runtime variable as a parameter to a later action (or to control contingent execution), information gathered by one action can affect the agent's subsequent behavior. Inside an effect, runtime variables (syntactically identified with a leading an exclamation point, *e.g.* *!tv*) can appear as terms or as truth values. For example, `ping twain` has the effect **observe**(machine.alive(`twain`), *!tv*), *i.e.* determining whether it is true or false that the machine named `twain` is alive, and `wc myfile` has the effect **observe**(word.count(`myfile`, *!word*)), *i.e.* determining the number of words in `myfile`.

Before we define individual effects, we discuss what it means to execute an action, with all its effects. Let $\mathrm{EFF}(E, a, s)$ denote the fact that E becomes true after action a is executed in s, let π^a be the precondition of action a, and let ε^a be the effects. An action's effects will only be realized if the action is executed when its preconditions are satisfied. Furthermore, the agent always knows when it executes an action, and it knows the effects of that action. Following Moore (Moore 1985):

$$\forall s.\mathrm{GOAL}(\pi^a, s, \{\}) \Rightarrow \forall s''.[K(s'', \mathrm{DO}(a, s)) \Leftrightarrow \exists s'.\mathrm{K}(s', s) \wedge s'' = \mathrm{DO}(a, s') \wedge \mathrm{EFF}(\varepsilon^a, a, s)] \quad (9)$$

The fact that the agent knows the effects of a doesn't imply that effects are always certain. As we discuss in Section 3.3, actions with conditional effects can result in uncertainty.

We now define the semantics of **observe** in terms of primitive situation calculus expressions:

$$\mathrm{EFF}(\textbf{observe}(P, \mathrm{T}), a, s) \stackrel{\mathrm{def}}{=} \forall s'.\mathrm{K}(s', \mathrm{DO}(a, s)) \Rightarrow \exists s_i.\mathrm{K}(s_i, s) \wedge s' = \mathrm{DO}(a, s_i) \wedge P(s_i) \quad (10)$$

$$\mathrm{EFF}(\textbf{observe}(P, tv), a, s) \stackrel{\mathrm{def}}{=} \forall s'.\mathrm{K}(s', \mathrm{DO}(a, s)) \Rightarrow \exists s_i.\mathrm{K}(s_i, s) \wedge s' = \mathrm{DO}(a, s_i) \wedge (P(s_i) \Leftrightarrow P(s)) \quad (11)$$

In other words if action a has an **observe** effect and is executed in situation s, then in the resulting situation, the agent knows more about the value that P had in s. For example, if in s the agent observes that the sky is blue, we would say that in situation $s' = \mathrm{DO}(\texttt{look}, s)$, the agent knows that the sky was blue in situation s. The double use of the K operator in Equations 9 and 10 is a trifle redundant given only a single observational effect. Indeed, if we assume positive introspection (*i.e.* K is transitive), as in the S4 logic, the resulting equation can be greatly simplified.

```
action ls(d)
    precond:  satisfy(current.shell(csh)) ∧
              satisfy(protection(d, readable)) ∧
    effect:   ∀ !f when in.dir(!f, d)
                ∃ !p, !n
                  observe(in.dir(!f, d)) ∧
                  observe(pathname(!f, !p)) ∧
                  observe(name(!f, !n))
```

Figure 2: **UNIX action schema.** The SADL `ls` action (UNIX `ls -a`) to list all files in the a directory.

However, in more complex effects, we wish to distinguish between the agent knowing that the effect as a whole took place, and knowing the value of a single fluent.

SADL supports universally quantified run-time variables. By nesting universal and existential quantifiers, SADL can model powerful sensory actions that provide several pieces of information about an unbounded number of objects. For example, `ls -a`, (Figure 2), reports several facts about each file in the current directory. The universal quantifier indicates that, at execution time, information will be provided about *all* files *!f* which are in directory *d*. Since the value of *!f* is observed, quantification uses a run-time variable. The nested existential quantifier denotes that each file has a *distinct* filename and pathname. The conditional **when** restricts the files sensed to those in directory *d*. The fact that the in.dir relation appears in two places may seem odd, but as we shall explain, the first use of in.dir refers to the actual situation s, whereas the second refers to the agent's knowledge (*i.e.*, all possible situations).

It is useful to note that after executing `ls -a tex`, the agent not only knows all files in `tex`; she knows that she knows all files (*i.e.*, she has closed world knowledge on the contents of `tex`). Because of the ∀ in the effects of `ls`, and since she knows the effects of `ls`, the agent can infer closed-world knowledge. Such inference would be costly if it were done using first-order theorem-proving in the situation calculus. We have devised efficient algorithms for doing this reasoning, which we describe in (Etzioni, Golden, & Weld 1997; 1994).

The translation of ∀ effects into the situation calculus is straightforward (Other logical operators follow the same form):

$$\mathrm{EFF}(\forall \vec{x}.E, a, s) \stackrel{\mathrm{def}}{=} \forall \vec{x}.\mathrm{EFF}(E, a, s) \quad (12)$$

This definition of ∀ effects may seem anticlimactic. The magic, however, stems from the way in which **when** introduces secondary preconditions; these are required for ∀ effects, where the **when** clause restricts the universe of discourse to a finite set, and indicates

precisely the range of the quantifier.

3.2 CONDITIONAL EFFECTS

A secondary precondition, *i.e.* one associated with an effect (Pednault 1989), defines the conditions under which action execution will achieve that effect. Unlike primary preconditions, secondary preconditions need not be true for the action to be executed. If p is the secondary precondition of effect e, then the resulting conditional effect is defined as:

$$\text{EFF}(\textbf{when}(p,e),a,s) \stackrel{\text{def}}{=} \text{GOAL}(p,s,\{\}) \Rightarrow \text{EFF}(e,a,s)$$

We use GOAL in our definition of **when**, but we have only defined GOAL for annotations **satisfy**, **hands-off** and **initially**. How should we define **when** preconditions? Since they need to hold, if at all, when the action is executed, they are different from **initially** preconditions. But **satisfy** requires that the agent know that the condition is true, which would lead to the faulty conclusion that the effect only occurs if the agent *knows* that the secondary preconditions hold. So we add a new type of precondition, without any annotation at all, to represent conditions that must hold at the time of execution, with or without knowledge of the agent:

$$\text{GOAL}(P,s_0,\{a\}_1^n) \stackrel{\text{def}}{=} P(s_n) \tag{13}$$

This ensures that whether the effects occur depends only on the state of the world. It also makes it clear what is being quantified over in `ls`: The files *really* in d, at the time of execution.

3.3 UNCERTAIN EFFECTS

In some cases, executing actions with causal effects can decrease the agent's knowledge about the world. SADL provides two ways of encoding these actions: as conditional effects whose secondary precondition is unknown, or by explicitly specifying the `U` truth value. As an example of the former, executing `rm tex/*` deletes all writable files in `tex`; if the agent doesn't know which files are writable, then she won't know which files remain in `tex` even if she knew the contents before executing the action. As an example of explicit creation of uncertainty, we encode `compress myfile` with the effect $\forall n$ **cause** `(size (myfile, n), U)`.[7] We define causal effects for `T` and `U` truth values as follows:

$$\text{EFF}(\textbf{cause}(P,\text{T}),a,s) \stackrel{\text{def}}{=} P(\text{DO}(a,s)) \tag{14}$$

$$\text{EFF}(\textbf{cause}(P,\text{U}),a,s) \stackrel{\text{def}}{=}$$
$$Unk_P(a,\text{DO}(a,s)) \Leftrightarrow P(\text{DO}(a,s)) \tag{15}$$

[7] In principle, we could represent all uncertain effects as conditional effects with unknown preconditions, but doing so would be cumbersome. However, we define the *semantics* of uncertain effects in precisely this manner.

where, Unk_P is a predicate such that

$$\neg KNOW(Unk_P(a),\text{DO}(a,s)) \wedge$$
$$\neg KNOW(\neg Unk_P(a),\text{DO}(a,s)) \tag{16}$$

In other words, we represent an uncertain effect as a deterministic function of hidden state. $Unk_P(a)$ denotes a *unique* unknown predicate, which represents the hidden state responsible for the change in truth value of P. It must be unique to avoid biasing correlation of independent unknown effects.

It is clear from the above definition how a **cause** effect may make P unknown. What may not be clear is how a **cause** effect can make P known. In fact, it wouldn't, if not for the fact that the agent knows all the effects of an action (Equation 9). However, knowledge of a conditional effect does not necessarily mean knowledge of the consequent. For example, if an agent executes `compress myfile`, she only knows that *if* she had write permission prior to executing `compress`, then `myfile` is compressed afterward.

4 TEMPORAL PROJECTION & REGRESSION

We have discussed the function DO, which maps a situation and an action (or sequence of actions) to a new situation, but we haven't yet said how the two situation terms relate to each other. If $s' = \text{DO}(\{a\}_1^n,s)$, we want to answer the following questions.

- Progression: What can we say about s', given knowledge of the conditions that hold in s?
- Regression: What must be true in s, to guarantee some desired condition in s'?

We treat each in turn.

4.1 PROJECTION & THE FRAME PROBLEM

The definitions for preconditions and effects that we have given are insufficient to solve the temporal projection problem. SADL effects only list fluents that an action affects, but what about fluents it doesn't affect? Explicitly stating everything that *doesn't* change would be tedious — this is the well-known frame problem. The standard approach to the frame problem, and the one we adopt, is to make the STRIPS assumption: anything not explicitly said to change remains the same. To fully specify the SADL semantics, it is necessary to express the STRIPS assumption in terms of the situation calculus. We use the formulation introduced in (Reiter 1991), and augmented in (Scherl & Levesque 1993) to account for sensing actions. This strategy consists of providing a formula for each fluent, called a *successor state axiom*, that specifies the

value of the fluent in terms of 1) the action executed, and 2) the conditions that held before the action was executed. By quantifying over actions, we can produce a single, concise formula for each fluent that includes only the relevant information.

Specifying update axioms for each fluent independently requires fluents to be logically independent of each other, so disjunction is not allowed. Effects consist of conjunctions of terms, each term being equivalent to one of the following

$$\textbf{when } \gamma^{\mathrm{T}}_P(a) \textbf{ cause}(P, \mathrm{T}) \quad (17)$$

$$\textbf{when } \gamma^{\mathrm{F}}_P(a) \textbf{ cause}(P, \mathrm{F}) \quad (18)$$

$$\textbf{when } \gamma^{\mathrm{U}}_P(a) \textbf{ cause}(P, \mathrm{U}) \quad (19)$$

$$\textbf{when } \kappa^{tv}_P(a) \textbf{ observe}(P, tv) \quad (20)$$

where a is an action and P is a fluent, which may contain universally quantified variables or constants,[8] and $\gamma^{tv}_P(a)$ and $\kappa^{tv}_P(a)$ represent arbitrary goal expressions.[9] For example, if `compress tex/*` changes the size of all writable files in directory `tex`, then $\gamma^{\mathrm{U}}_{\mathrm{size}(f)}(\texttt{compress tex/*}) = \mathrm{indir}(f, \texttt{tex}) \wedge \mathrm{writable}(f)$. Clearly, all actions can be represented by specifying the γ and κ preconditions for each fluent in the domain theory. If a has a non-conditional effect, **cause**(P, tv), then $\gamma^{tv}_P(a) = \mathrm{T}$. We can express the fact that action a doesn't affect P at all by saying $\forall tv.\gamma^{tv}_P(a) = \mathrm{F}$. We don't list **observe**(P, T) above, since it is subsumed by the conjunction **observe**(P, v) $\wedge\, v = \mathrm{T}$ (similarly for F).

Given these definitions, we can state the conditions under which an action changes or preserves a fluent's truth value. Following Pednault (Pednault 1986), we define Σ^a_φ to be the conditions under which an executable action a will establish φ, and Π^a_φ to be the conditions under which a will preserve φ. We have the following establishment conditions:

$$\Sigma^a_\varphi \Leftrightarrow \gamma^{\mathrm{T}}_\varphi(a) \vee (Unk_\varphi(a) \wedge \gamma^{\mathrm{U}}_\varphi(a)) \quad (21)$$

$$\Sigma^a_{\neg\varphi} \Leftrightarrow \gamma^{\mathrm{F}}_\varphi(a) \vee (\neg Unk_\varphi(a) \wedge \gamma^{\mathrm{U}}_\varphi(a)) \quad (22)$$

where $Unk_\varphi(a)$ is the unknown predicate introduced in Equations 15 and 16. The presence of an effect with a U truth value will make φ true or false, depending on the value of $Unk_\varphi(a)$. Since $Unk_\varphi(a)$ is unknown by definition, effects with U truth values aren't generally useful for goal establishment. We also have the following preservation conditions:

$$\Pi^a_\varphi \Leftrightarrow \neg\gamma^{\mathrm{F}}_\varphi(a) \wedge (\neg Unk_\varphi(a) \vee \neg\gamma^{\mathrm{U}}_\varphi(a)) \quad (23)$$

$$\Pi^a_{\neg\varphi} \Leftrightarrow \neg\gamma^{\mathrm{T}}_\varphi(a) \wedge (Unk_\varphi(a) \vee \neg\gamma^{\mathrm{U}}_\varphi(a)) \quad (24)$$

[8] Including variables that will resolve to constants.

[9] with the restriction that effects must be consistent, so, for example, $\gamma^{\mathrm{T}}_P(a) \wedge \gamma^{\mathrm{F}}_P(a)$ must always be false.

For each fluent, we can then generate an expression that specifies precisely when it is true or false, by quantifying over actions. For each fluent P, there is a *successor state axiom*, which combines update axioms and frame axioums for P. The successor state axioms are straightforward statements of the STRIPS assumption: a fluent is true if and only if it was made true, or it was true originally and it wasn't made false:

$$\mathrm{GOAL}(\pi^a, s, \{\}) \Rightarrow [P(\mathrm{DO}(a, s)) \Leftrightarrow \Sigma^a_P(s) \vee P(s) \wedge \Pi^a_P(s)] \quad (25)$$

Similarly, there is a successor state axiom for K.

$$\begin{aligned}\mathrm{GOAL}(\pi^a, s, \{\}) \Rightarrow [K(s'', \mathrm{DO}(a, s)) \Leftrightarrow \exists s'.\mathrm{K}(s', s) \\ \wedge (s'' = \mathrm{DO}(a, s')) \wedge \\ \forall P.([\kappa^v_P(a, s) \wedge P(s)] \Rightarrow [P(s') \wedge v =_{s'} \mathrm{T}] \wedge \\ ([\kappa^v_P(a, s) \wedge \neg P(s)]) \Rightarrow [\neg P(s') \wedge v =_{s'} \mathrm{F}])]\end{aligned} \quad (26)$$

We have stated this formula in second-order logic, but only because the formula depends on all of the actual fluents in the domain theory. Given any specific domain, this second-order formula could be replaced with an equivalent first-order formula by replacing P with each fluent in the domain.

The above definition only specifies when information is gained, and seems to say nothing about when it is lost. However, information loss is indeed accounted for, through the successor state axiom for P. If P becomes true in some situations accessible from s, and false in others, then by definition, P is unknown. For example, `compress myfile` compresses `myfile` if it is writable. If it is unknown whether `myfile` is writable, then in some accessible worlds, `myfile` is writable and will be compressed. In other worlds, `myfile` is not writable and won't be compressed. The result is that it becomes unknown whether `myfile` is compressed. Similarly, if P was known previously and not changed, then by the successor state axioms for P and K, P will continue to be known. (Scherl & Levesque 1993).

The above formula correctly describes how K changes, but it is a little unwieldy if what we want to know about is KNOW(φ). Intuitively, KNOW(φ) becomes true if φ is known to become true, or φ is observed. Additionally, φ continues to be known true until it possibly becomes false. The following formulas follow from the successor state axioms for φ and K.

$$\Pi^a_{\mathrm{KNOW}(\varphi)} \Leftrightarrow \mathrm{KNOW}(\Pi^a_\varphi) \quad (27)$$

$$\Sigma^a_{\mathrm{KNOW}(\varphi)} \Leftrightarrow KNOW(\gamma^{\mathrm{T}}_\varphi(a)) \vee (\kappa^{tv}_\varphi(a) \wedge \varphi \wedge \Pi^a_{\mathrm{KNOW}(\varphi)})^{10} \quad (28)$$

[10] The additional requirement $\Pi^a_{\mathrm{KNOW}(\varphi)}$ may come as a surprise, since an action that simultaneously observes φ

4.2 REGRESSION

Most modern planners build plans using goal regression — starting with a goal and successively adding actions that achieve either part of the goal or preconditions of previously added actions. Once no preconditions remain that aren't true in the initial state, the plan is complete. It is therefore useful to have a formal specification of what conditions must be true for a given action sequence to achieve a given goal. Let a^{-1} be a *regression operator* for action a. $a^{-1}(\varphi)$ is a condition that, if true immediately before the execution of a, results in φ being true after a is executed. We define $(\{a\}_1^n)^{-1}(\varphi)$ to be $a_n{}^{-1}(a_{n-1}{}^{-1}(\ldots(a_1{}^{-1}(\varphi))))$. Naturally, regression on an action sequence of zero length is the identity function: $\{\}^{-1}(\varphi) = \varphi$.

Let α be an axiomatization of the initial conditions, and let Γ be some goal expression. The objective of planning is to produce an executable sequence of actions, $\{a\}_1^n$, such that $\alpha \models (\{a\}_1^n)^{-1}(\Gamma)$. We discuss executability in Section 4.3.

We specify regression operators for **satisfy**, **initially** and **hands-off** goals below. Since some conditions could be true in the initial state, we also must specify when a condition is true after executing a plan of zero length. Since **initially** indicates something that must be true before the plan is executed, and **satisfy** indicates things true afterwards, it follows that if there is no plan, then **initially** and **satisfy** have the same interpretation: For all φ,

$$\alpha \models \text{KNOW}(\varphi, S_0) \Leftrightarrow \{\}^{-1}(\textbf{initially}(\varphi)) = \text{T} \quad (29)$$

$$\alpha \models \text{KNOW}(\varphi, S_0) \Leftrightarrow \{\}^{-1}(\textbf{satisfy}(\varphi)) = \text{T} \quad (30)$$

hands-off is always true in the initial state, since it can only be violated by changing the proscribed fluent:

$$\{\}^{-1}(\textbf{hands-off}(\varphi)) = \text{T} \quad (31)$$

We now consider how to regress a SADL goal formula through an action. A goal **satisfy**(φ) is achieved if the agent knows that φ is true; *i.e.*, φ just became true, was just observed to be true, or was previously known to be true and wasn't subsequently affected. The first two conditions are captured by $\Sigma^a_{\text{KNOW}(\varphi)}$. The latter holds when **satisfy**(φ) held in the previous state, and knowledge of φ was preserved:

$$a^{-1}(\textbf{satisfy}(\varphi)) = \Sigma^a_{\text{KNOW}(\varphi)} \vee (\textbf{satisfy}(\varphi) \wedge \Pi^a_{\text{KNOW}(\varphi)}) \quad (32)$$

A **hands-off** goal holds if the state of φ always remains the same as it was in the initial state. **hands-off**(φ) doesn't forbid actions that *affect* φ — just actions that *change* φ. For example, an action `compress myfile` doesn't violate the goal **hands-off**(compressed(`myfile`)) if `myfile` was already compressed initially.[11]

$$a^{-1}(\textbf{hands-off}(\varphi)) = (\Pi^a_{\neg\varphi} \vee \textbf{initially}(\varphi)) \wedge (\Pi^a_{\varphi} \vee \textbf{initially}(\neg\varphi)) \wedge \textbf{hands-off}(\varphi) \quad (33)$$

initially(φ) is satisfied after action a if it was already satisfied, or if φ was observed by action a, and wasn't affected by any previous actions. Unlike other goals, we are interested in the *first* time point at which an **initially** goal is achieved, as opposed to the last. The disjunct **initially**(φ) ensures that the first occurrence is considered, because it is always regressed back.

$$a^{-1}(\textbf{initially}(\varphi)) = \textbf{initially}(\varphi) \vee (\kappa^{tv}_{\varphi}(a) \wedge \varphi \wedge \textbf{hands-off}(\varphi)) \quad (34)$$

This definition doesn't rule out using destructive sensing actions. All that matters is that φ be undisturbed *before* it is sensed. It's fine if the act of sensing the value of φ itself affects φ.

Unannotated preconditions merely need to be satisfied in the final state, and it isn't necessary that they be known true.

$$a^{-1}(\varphi) = \Sigma^a_{\varphi} \vee (\varphi \wedge \Pi^a_{\varphi}) \quad (35)$$

Logical operators are simply regressed back to the initial state, since their interpretation is the same across all situations, as detailed in (Pednault 1988).

With these definitions, we can show that regression is correct — that is, if the conditions returned by $a^{-1}(\Gamma)$ are true, and $\{a\}_1^n$ is successfully executed, then Γ will indeed be true.

Theorem 1 (Soundness of Regression) *Let $\{a\}_1^n$ be an executable action sequence. Let Γ be a goal formula, and let α be an axiomatization of the initial state, s_0. Then $\alpha \models (\{a\}_1^n)^{-1}(\Gamma) \Rightarrow GOAL(\Gamma, s_0, \{a\}_1^n)$*

We believe that the reverse is also true — *i.e.*, if Γ is true after $\{a\}_1^n$ is executed, then $a^{-1}(\Gamma)$ must have been true.

and causes φ to become false or unknown would seem to violate our rule against inconsistent actions. However, such effects aren't inconsistent, since the observation pertains to situation s, whereas the update is to situation DO(a, s). Such *destructive sensing actions* are commonplace. By the Heisenberg Uncertainty Principle, they are inevitable, but examples can be found in macroscopic domains as well. Biologists find out the number of insects living in a tree by placing containers under the tree and then fogging the tree with poison. The number of insects that fall into the containers provides an estimate of the number that were originally living there.

[11] This is a departure from UWL's notion of **hands-off**, in which the `compress` would be a violation. However, uncompressing the file and then recompressing it does violate the goal, since the uncompress changes the fluent.

4.3 EXECUTABILITY

Regression operators alone only tell part of the story about when an action, or sequence of actions, can achieve a goal. $a^{-1}(\varphi)$ consists of the conditions under which a will achieve φ *assuming it is successfully executed.* So to ensure that a brings about φ, we must also ensure that a can be executed. Action a is executable in situation s iff the preconditions of a are true in s. A sequence of actions, $\{a\}_1^n$, is executable in s iff a_1 is executable in s, a_2 is executable in $\mathrm{DO}(a_1, s)$, a_3 is executable in $\mathrm{DO}(a_2, \mathrm{DO}(a_1, s))$, and so on.

5 EXPRESSIVENESS

Although SADL is appropriate for any Markov domain (*e.g.*, transportation logistics, manufacturing, mobile robotics, *etc.*), the language is best at modeling domains with accurate (low noise) sensors. We have concentrated our efforts on UNIX and the Internet, encoding hundreds of commands. Examples of sensory actions include `finger`, `wc`, `grep`, the `netfind` and `inspec` Internet sites, and actions to traverse the Web; causal actions include `cp`, `rm`, and `compress`. Universal quantification allows us to model actions that return an unbounded amount of information, such as `ls` (Figure 2).

As an illustration, consider the goal, Γ, of finding a file named `old` and renaming it to `new`: $\Gamma = \exists f.\mathbf{initially}(\mathrm{name}(f, \mathtt{old})) \wedge \mathbf{satisfy}(\mathrm{name}(f, \mathtt{new}))$. Recall that this goal is inexpressible in UWL. It can be achieved by executing `ls` in various directories until the desired file is found, and then executing `mv` to change the name to `new`. There is no single action sequence that will work in all situations, because the location of `old` is not necessarily known. Let's assume that `old` resides in the directory `tex`, and that its location is unknown. We also assume that the agent knows that `tex` is readable, and that current.shell(`csh`) is true. The shortest possible action sequence that would achieve the goal is `ls tex` then `mv tex/old tex/new`. For brevity, we abbreviate these actions as `ls` and `mv`, respectively. We show that this action sequence is executable, and that it achieves the goal.

For the sake of this example, we won't consider `mv` in its full glory. Rather, we assume a simplified version of `mv`, with the precondition $\pi^{\mathtt{mv}} = \mathbf{satisfy}(\mathrm{name}(f, \mathtt{old})) \wedge \mathbf{satisfy}(\mathrm{in.dir}(f, \mathtt{tex}))$, and the single effect $\varepsilon^{\mathtt{mv}} = \mathbf{cause}(\mathrm{name}(f, \mathtt{new}))$. This representation ignores many details, such as whether `tex` is writable, `old` is readable, there is already a file named `new`, etc.

To show that the plan is executable, we must first show that the preconditions of `ls` hold in S_0, *i.e.*, $S_0 \models \pi^{\mathtt{ls}}$, and then show that the preconditions of `mv` hold after `ls` is executed, *i.e.*, $\mathrm{DO}(\mathtt{ls}, S_0) \models \pi^{\mathtt{mv}}$. To show that the plan achieves the goal, we need show that $\mathrm{DO}(\mathtt{mv}, \mathrm{DO}(\mathtt{ls}, S_0)) \models \Gamma$. We use regression to show that these results hold.

We first regress the two conjuncts of Γ through `mv`. `mv` achieves the **satisfy** goal, with no secondary preconditions: $\mathtt{mv}^{-1}(\mathbf{satisfy}(\mathrm{name}(f, \mathtt{new}))) \Leftarrow \Sigma^{\mathtt{mv}}_{\mathrm{KNOW}(\mathrm{name}(f,\mathtt{new}))} \Leftarrow \mathrm{KNOW}(\gamma^{\mathrm{T}}_{\mathrm{name}(f,\mathtt{new})}(\mathtt{mv})) \Leftarrow \mathrm{KNOW}(\mathrm{T}) \Leftarrow \mathrm{T}$.

`mv` has no effect on the **initially** goal: $\mathtt{mv}^{-1}(\mathbf{initially}(\mathrm{name}(f, \mathtt{old}) \Leftarrow \mathbf{initially}(\mathrm{name}(f, \mathtt{old}))$.

Now we regress $\mathtt{mv}^{-1}(\Gamma) \wedge \pi^{\mathtt{mv}}$ through `ls`. That is, we regress $\mathbf{initially}(\mathrm{name}(f, \mathtt{old})) \wedge \mathbf{satisfy}(\mathrm{name}(f, \mathtt{old})) \wedge \mathbf{satisfy}(\mathrm{in.dir}(f, \mathtt{tex}))$. We regress the first two conjuncts through `ls`. The final conjunct, $\mathbf{satisfy}(\mathrm{in.dir}(f, \mathtt{tex}))$, follows the same pattern.

The action `ls tex` has the effect $\forall f \exists n$ $\mathbf{observe}(\mathrm{name}(f, n)) \wedge \mathbf{observe}(\mathrm{in.dir}(f, \mathtt{tex}))$, with the secondary precondition $\mathrm{in.dir}(f, \mathtt{tex})$. This precondition does not require knowledge on the part of the agent. So $\mathtt{ls}^{-1}(\mathbf{initially}(\mathrm{name}(f, \mathtt{old})) \wedge \mathbf{satisfy}(\mathrm{name}(f, \mathtt{old}))) \Leftarrow \Sigma^{\mathtt{ls}}_{\mathrm{KNOW}(\mathrm{name}(f,\mathtt{old}))} \wedge \mathbf{hands\text{-}off}(\mathrm{name}(f, \mathtt{old})) \wedge \Pi^{\mathtt{ls}}_{\neg\mathrm{name}(f,\mathtt{old})} \Leftarrow \kappa^{\mathrm{T}}_{\mathrm{KNOW}(\mathrm{name}(f,\mathtt{old}))}(\mathtt{ls}) \wedge \mathrm{KNOW}(\Pi^{\mathtt{ls}}_{\mathrm{name}(f,\mathtt{old})}) \wedge \mathrm{name}(f, \mathtt{old}) \wedge \mathbf{hands\text{-}off}(\mathrm{name}(f, \mathtt{old})) \wedge \neg\gamma^{\mathrm{T}}_{\mathrm{name}(f,\mathtt{old})}(\mathtt{ls}) \wedge \neg\gamma^{\mathrm{U}}_{\mathrm{name}(f,\mathtt{old})}(\mathtt{ls}) \Leftarrow \mathrm{in.dir}(f, \mathtt{tex}) \wedge \mathrm{KNOW}(\neg\gamma^{\mathrm{T}}_{\mathrm{name}(f,\mathtt{old})}(\mathtt{ls})) \wedge \neg\gamma^{\mathrm{U}}_{\mathrm{name}(f,\mathtt{old})}(\mathtt{ls})) \wedge \mathrm{name}(f, \mathtt{old}) \Leftarrow \mathrm{in.dir}(f, \mathtt{tex}) \wedge \mathrm{KNOW}(\mathrm{T}) \wedge \mathrm{name}(f, \mathtt{old})$.

This last formula is entailed by S_0. All that remains is to show that $S_0 \models \{\}^{-1}(\pi^{\mathtt{ls}})$. By the definition of $\{\}^{-1}$ for **satisfy** goals, that follows iff $S_0 \models \mathrm{KNOW}(\mathrm{current.shell}(\mathtt{csh})) \wedge \mathrm{KNOW}(\mathrm{protection}\ \mathtt{tex}, \mathtt{readable})$, which is true by assumption.

6 TRACTABILITY

SADL is implemented by XII (Golden, Etzioni, & Weld 1994; 1996), a partial-order planner whose performance is comparable to the UCPOP/SNLP family of classical planners. We analyze its performance in terms of the refinement paradigm described in (Kambhampati, Knoblock, & Yang 1995) — XII has three refinement operations: goal establishment, conflict resolution and action execution. Goal establishment involves possibly adding an action to the plan, and adding an *interval protection constraint* (IPC) to prevent the goal from being clobbered. In SADL, there are three possible intervals to consider. If s_p is the situation in which the action will be executed, and s_c is the situation in which the goal is to be fulfilled, the intervals are $[s_{p+1}, s_c]$, $[s_0, s_p]$ or $[s_0, s_c]$, corresponding

to **satisfy**, **initially** or **hands-off**, respectively. Ensuring that no actions violate the IPC requires $O(n)$ time, where n is the number of steps in the plan, but maintaining a consistent ordering of actions requires $O(n^2)$ time. Conflict resolution and action execution also take $O(n^2)$ time. In contrast, note that goal establishment and conflict resolution are undecidable in the situation calculus.

7 RELATED WORK

(McCarthy & Hayes 1969) first argued that an agent needs to reason about its ability to perform an action. (Moore 1985) devised a theory of knowledge and action, based on a variant of the situation calculus with possible-worlds semantics. He provided an analysis of knowledge preconditions, which we discussed earlier, and information-providing effects. (Morgenstern 1987) generalized Moore's results to express partial knowledge that agents have about the knowledge of other agents (e.g. "John knows what Bill said"), using a substantially more expressive logic, which is syntactic rather than modal. (Davis 1993) extended Moore's theory to handle contingent plans, though, like Moore, he doesn't discuss actions with indeterminate effects. (Levesque 1996) offers an elegant theory of when a plan, with conditionals and loops, achieves a satisfaction goal in the presence of incomplete information. However, Levesque doesn't discuss knowledge goals, and his sensory actions can return only T or F, and can't change the state of the world. (Goldman & Boddy 1996) present a clean language for contingent plans with context-dependent effects and nondeterminism. However, like Levesque, they don't allow variables in sensing actions: possible outcomes are represented as a disjuction. (Shoham 1993) presents a language, with explicit time, for representing beliefs and communication among multiple agents. Agents can request other agents to perform actions, which can include (nested) communicative actions, but not arbitrary goals. A discrete temporal logic, without $\forall$, is used to represent beliefs. PRS (Ingrand *et al.* 1996) is a procedural language that can represent a similar class of goals as SADL, but lacks temporal goals such as **initially**. PRS has annotation **achieve** corresponding to SADL **satisfy**, **preserve** corresponding to **hands-off**, and **test** corresponding to **satisfy+hands-off**, as well as several procedural constructs that have no corresponding terms in the declarative SADL language.

Partially-observable Markov Decision Processes (Monahan 1982; Cassandra, Kaebling, & Littman 1994) provide an elegant representation of sensing actions and actions with uncertain outcomes in Markov domains. However, they don't lend themselves to efficient algorithms. With few exceptions, such as (Bacchus, Boutilier, & Grove 1995), work in MDPs assumes that reward functions (goals) are Markov as well, so temporal goals like **initially** are inexpressible.

A number of contingent planning systems have introduced novel representations of uncertainty and sensing actions. Warplan-C (Warren 1976) tags actions as conditional, meaning they have two possible outcomes: P or $\neg P$. C-BURIDAN (Kushmerick, Hanks, & Weld 1995; Draper, Hanks, & Weld 1994) uses a probabilistic action language that can represent conditional, observational effects, including noisy sensors, and effects that cause information loss. Unlike SADL, the C-BURIDAN language is propositional, and makes no distinction between knowledge goals and goals of satisfaction. C-BURIDAN and Cassandra (Pryor & Collins 1996) (and WCPL (Goldman & Boddy 1996)) can represent and reason with uncertain outcomes of actions as disjunctions, allowing them to deal with correlations between multiple unknown variables (e.g. either it is raining and Fido is wet, or it is sunny and Fido is dry). By using the `U` truth value, SADL gives up the ability to represent these correlations (*i.e.* as far as the agent knows, it is raining and fido is dry). However, reasoning with `U` truth values is more efficient than the possible-worlds representation used to handle disjunction. CNLP (Peot & Smith 1992), like SADL, uses a three-valued logic to represent uncertainty. Another limitation of these other languages is an inability to represent actions, like `ls` that return information about an unbounded number of objects.

8 CONCLUSIONS

We introduced SADL, a language for representing sensing actions and information goals, which embodies the lessons learned during four years of building and debugging Internet Softbot domain theories: 1) Since knowledge goals are temporal, SADL supports the temporal annotation **initially**. 2) In *Markov domains*, such as UNIX, knowledge preconditions for actions are inappropriate, but subgoaling to obtain knowledge is often necessary; SADL handles this paradox by eliminating knowledge preconditions from actions, but using secondary preconditions to clearly indicate when subgoaling to acquire knowledge could be useful. SADL is expressive enough to represent real-world domains, such as UNIX and the World Wide Web, yet restricted enough to be used efficiently by modern planning algorithms, such as XII (Golden, Etzioni, & Weld 1996).

Acknowledgements

Many thanks to Mark Boddy, Bob Doorenbos, Denise Draper, Oren Etzioni, Marc Friedman, Robert Goldman, Neal Lesh, Greg Linden, Mike Perkowitz, Rich Segal, Jonathan Shakes and Ellen Spertus for helpful comments. This research was funded in part by Office of Naval Research Grant N00014-94-1-0060, by National Science Foundation Grant IRI-9303461, by ARPA / Rome Labs grant F30602-95-1-0024, by a gift from Rockwell International Palo Alto Research, and by a Microsoft Graduate Fellowship

References

Bacchus, F.; Boutilier, C.; and Grove, A. 1995. Rewarding behaviors. In *Proc. 14th Nat. Conf. on AI.*

Cassandra, A. R.; Kaebling, L. P.; and Littman, M. L. 1994. Algorithms for partially observable markov decision processes. Technical report 94-14, Brown University, Providence, Rhode Island.

Davis, E. 1993. Knowledge preconditions for plans. Technical Report 637, NYU Computer Science Department.

Draper, D.; Hanks, S.; and Weld, D. 1994. A probabilistic model of action for least-commitment planning with information gathering. In *Proc. 10th Conf. on Uncertainty in Artifical Intelligence.*

Drummond, M. 1989. Situated control rules. In *Proceedings of the First International Conference on Knowledge Representation and Reasoning.*

Etzioni, O., and Weld, D. 1994. A softbot-based interface to the Internet. *CACM* 37(7):72–76.

Etzioni, O.; Hanks, S.; Weld, D.; Draper, D.; Lesh, N.; and Williamson, M. 1992. An approach to planning with incomplete information. In *Proc. 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning*, 115–125.

Etzioni, O.; Golden, K.; and Weld, D. 1994. Tractable closed-world reasoning with updates. In *Proc. 4th Int. Conf. on Principles of Knowledge Representation and Reasoning*, 178–189.

Etzioni, O.; Golden, K.; and Weld, D. 1997. Sound and efficient closed-world reasoning for planning. *Artificial Intelligence.* (To appear).

Golden, K.; Etzioni, O.; and Weld, D. 1994. Omnipotence without omniscience: Sensor management in planning. In *Proc. 12th Nat. Conf. on AI*, 1048–1054.

Golden, K.; Etzioni, O.; and Weld, D. 1996. Planning with execution and incomplete information. Technical Report 96-01-09, University of Washington, Department of Computer Science and Engineering. Available via FTP from `pub/ai/` at `ftp.cs.washington.edu`.

Goldman, R. P., and Boddy, M. S. 1996. Expressive Planning And Explicit Knowledge. In *Proc. 3rd Intl. Conf. on AI Planning Systems.*

Haddawy, P., and Hanks, S. 1993. Utility Models for Goal-Directed Decision-Theoretic Planners. Technical Report 93–06–04, Univ. of Washington, Dept. of Computer Science and Engineering. Submitted to *Artificial Intelligence.* Available via FTP from `pub/ai/` at `ftp.cs.washington.edu`.

Ingrand, F.; Chatila, R.; Alami, R.; and Robert, F. 1996. PRS: A high level supervision and control language for autonomous mobile robots. In *Proceedings of the 1996 IEEE International Conference On Robotics and Automation.*

Kambhampati, S.; Knoblock, C.; and Yang, Q. 1995. Planning as refinement search: A unified framework for evaluating design tradeoffs in partial order planning. *Artificial Intelligence* 76:167–238.

Kushmerick, N.; Hanks, S.; and Weld, D. 1995. An Algorithm for Probabilistic Planning. *Artificial Intelligence* 76:239–286.

Levesque, H. 1996. What is planning in the presence of sensing? In *Proc. 14th Nat. Conf. on AI.*

McCarthy, J., and Hayes, P. J. 1969. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence 4.* Edinburgh University Press. 463–502.

McCarthy, J. 1980. Circumscription - a form of non-monotonic reasoning. *Artificial Intelligence* 13(1,2):27–39.

Monahan, G. E. 1982. A survey of partially observable markov decision processes: Theory, models, and algorithms. *Management Science* 28(1):1–16.

Moore, R. 1985. A Formal Theory of Knowledge and Action. In Hobbs, J., and Moore, R., eds., *Formal Theories of the Commonsense World.* Norwood, NJ: Ablex.

Morgenstern, L. 1987. Knowledge preconditions for actions and plans. In *Proceedings of IJCAI-87*, 867–874.

Morgenstern, L. 1988. *Foundations of a Logic of Knowledge, Action, and Communication.* Ph.D. Dissertation, New York University.

Pednault, E. 1986. *Toward a Mathematical Theory of Plan Synthesis.* Ph.D. Dissertation, Stanford University.

Pednault, E. 1988. Synthesizing plans that contain actions with context-dependent effects. *Computational Intelligence* 4(4):356–372.

Pednault, E. 1989. ADL: Exploring the middle ground between STRIPS and the situation calculus. In *Proc. 1st Int. Conf. on Principles of Knowledge Representation and Reasoning*, 324–332.

Peot, M., and Smith, D. 1992. Conditional Nonlinear Planning. In *Proc. 1st Intl. Conf. on AI Planning Systems*, 189–197.

Pollack, M. 1992. The uses of plans. *Artificial Intelligence* 57(1).

Pryor, L., and Collins, G. 1996. Planning for contingencies: A decision-based approach. *Journal of Artificial Intelligence Research.*

Raiffa, H. 1968. *Decision Analysis: Introductory Lectures on Choices Under Uncertainty.* Addison-Wesley.

Reiter, R. 1991. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In Lifschitz, V., ed., *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy.* Academic Press. 359–380.

Scherl, R., and Levesque, H. 1993. The frame problem and knowledge producing actions. In *Proc. 11th Nat. Conf. on AI*, 689–695.

Shoham, Y. 1993. Agent-oriented programming. *Artificial Intelligence* 60(1):51–92.

Warren, D. 1976. Generating Conditional Plans and Programs. In *Proceedings of AISB Summer Conference*, 344–354.

Weld, D., and Etzioni, O. 1994. The first law of robotics (a call to arms). In *Proc. 12th Nat. Conf. on AI*, 1042–1047.

Wilkins, D. E. 1988. *Practical Planning.* San Mateo, CA: Morgan Kaufmann.

Williamson, M., and Hanks, S. 1994. Optimal planning with a goal-directed utility model. In *Proc. 2nd Intl. Conf. on AI Planning Systems.*

A New Algorithm for Generative Planning

Matthew L. Ginsberg
CIRL
1269 University of Oregon
Eugene, OR 97403-1269

Abstract

Existing generative planners have two properties that one would like to avoid if possible. First, they use a single mechanism to solve problems both of action selection and of action sequencing, thereby failing to exploit recent progress on scheduling and satisfiability algorithms. Second, the context in which a subgoal is solved is governed in part by the solutions to other subgoals, as opposed to plans for the subgoals being developed in isolation and then merged to yield a plan for the conjunction. We present a reformulation of the planning problem that appears to avoid these difficulties, describing an algorithm that solves subgoals in isolation and then appeals to a separate NP-complete scheduling test to determine whether the actions that have been selected can be combined in a useful way.

1 INTRODUCTION

One of the insights arising from the theoretical analysis of the complexity of domain-independent planning [1, 4, 14] is that planning, involving both the selection and sequencing of actions, is in general properly more difficult than scheduling, which involves sequencing problems in isolation. My goal in this paper is to introduce a new type of planning algorithm that responds to these results by separating the problem of action selection from that of action sequencing. By doing this, we can expect to incorporate into planners fast existing algorithms for solving NP-complete problems such as scheduling.

We will achieve this computational split by introducing a new planning language $\mathcal{L}$ that mediates between the action selection and action sequencing phases. Our intention is that a planner concerned with action selection in isolation could easily produce a sentence of $\mathcal{L}$ as its output, and that a scheduler could accept such a sentence as its input. In solving the Sussman anomaly, for example (Figure 1), the planner would realize that it needs to move block C off block A before moving block A onto block B, and also that it needs to move block B onto block C. The scheduling algorithm would then identify the ordering needed to actually achieve the goal (namely, that it is necessary to move blocks C, B and A in that order).

The language itself is introduced in Section 2, and we demonstrate its relevance to scheduling at a theoretical level in Section 3 by showing to be NP-complete the problem of deciding if a particular sentence in the language corresponds to a nonempty set of plans. We turn to planning in Section 4, developing an algorithm that generates instances of our planning language in response to a particular planning problem. This algorithm differs from existing partial-order causal link (POCL) planners in at least three fundamental ways.

First, it solves subgoals separately, merging the results to construct a plan for the overall goal itself. This property is recursive: Subgoals are solved separately at all levels, with the plans being merged to construct one that achieves the overall goal.

Second, it does no backtracking. Instead, the algorithm works by gradually refining a candidate set of plans that might achieve the goal. If a particular approach is tried for a goal g, and this approach involves developing plans to solve a subgoal g', any solutions found for the subgoal will remain available to the planner even if a different approach is eventually selected for g itself.

Finally, the planner needs to plan for distinct goals only once. Even if a particular goal g must be achieved as a precondition for each of two separate actions,

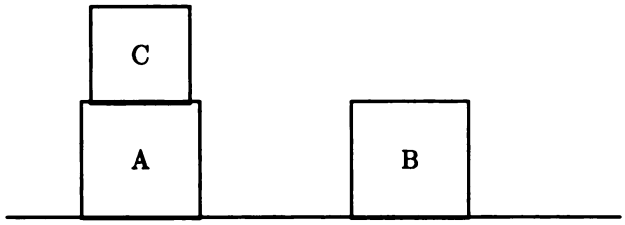


Figure 1: The Sussman anomaly: Get A on B on C

there will typically be only a single appearance of g in the planning search space.

We present an example in Section 5, where we also discuss the general computational properties of our procedure. Section 6 discusses an implementation of our work, and Section 7 contains a variety of concluding remarks, connecting our work to a variety of results that have appeared elsewhere. Proofs appear in an appendix.

2 THE PLAN LANGUAGE

We assume throughout that we have some fixed set of actions A.

Definition 2.1 *By a* linear plan *we will mean a finite sequence of elements of A. Such sequences will typically be denoted $\langle a_1, \ldots, a_n \rangle$.*

Of greater interest to us than specific linear plans will be *sets* of such plans. To this end, we define a language describing those sets in which we will be interested:

Definition 2.2 *The* plan language $\mathcal{L}$ *is the smallest set of sentences that contains the distinguished elements $\emptyset$ and $[\,]$ along with every element of A, and is closed under three binary operators $\cup$, $|$ and π.*

The set of plans $\emptyset$ is, as usual, the empty set. The plan $[\,]$ is intended to correspond to the sequence of no actions, and $a \in A \subseteq \mathcal{L}$ is the singleton plan where action a alone is taken. The operator $P|Q$ concatenates the plan Q to the end of the plan P, and $\pi(P,Q)$ is intended to capture the frame axiom in our setting. Let us give the formal definition of π first and then explain its meaning:

Definition 2.3 *Let $P \in \mathcal{L}$ be a set of plans. Membership in P is defined recursively as follows:*

1. $\langle a_1, \ldots, a_n \rangle \notin \emptyset$.
2. $\langle a_1, \ldots, a_n \rangle \in [\,]$ *if and only if* $n = 0$.
3. $\langle a_1, \ldots, a_n \rangle \in a$ *for an action a if and only if* $n = 1$ *and* $a_1 = a$.
4. $\langle a_1, \ldots, a_n \rangle \in P|Q$ *if and only if for some* $i \leq n$, $\langle a_1, \ldots, a_i \rangle \in P$ *and* $\langle a_{i+1}, \ldots, a_n \rangle \in Q$.
5. $\langle a_1, \ldots, a_n \rangle \in P \cup Q$ *if and only if* $\langle a_1, \ldots, a_n \rangle \in P$ *or* $\langle a_1, \ldots, a_n \rangle \in Q$.
6. *To determine if* $\langle a_1, \ldots, a_n \rangle \in \pi(P,Q)$, *let i be the greatest integer such that* $\langle a_1, \ldots, a_i \rangle \in P \cup Q$. *Then* $\langle a_1, \ldots, a_n \rangle \in \pi(P,Q)$ *if and only if such an i exists and* $\langle a_1, \ldots, a_i \rangle \notin Q$.

It is presumably only the last of the above clauses that requires explanation. To understand it, suppose that we have some goal that is established by the plans in P, but disestablished by the plans in Q (clobbered, in Chapman's more picturesque terms [2]). Now the goal will hold after an action sequence $\langle a_1, \ldots, a_n \rangle$ has been executed if and only if it is established by some initial subsequence of $\langle a_1, \ldots, a_n \rangle$ and not subsequently clobbered – in other words, if the longest subsequence $\langle a_1, \ldots, a_i \rangle$ that either establishes or disestablishes the goal in fact establishes it. By writing the definition to require that the action a_i not disestablish the goal, we cater to the possibility that $P \cap Q \neq \emptyset$.

Lemma 2.4 $\pi(P_1, Q_1) \subseteq \pi(P_2, Q_2)$ *if* $P_1 \subseteq P_2$ *and* $Q_1 \supseteq Q_2$. ■

We now make the following notational definition:

Definition 2.5

1. $U \equiv \pi([\,], \emptyset)$.
2. $\neg P \equiv \pi(U, P)$.
3. $P \cap Q \equiv \neg(\neg P \cup \neg Q)$.

Lemma 2.6 *U is the set of all linear plans. Negation and intersection have their conventional meanings.*

In the Sussman anomaly, for example, the goal of getting A on B is established by moving A to B, which succeeds in at most the plans given by $U|\mathtt{move}(A,B)$. Moving C to B succeeds for at most the plans in $U|\mathtt{move}(C,B)$. For the preconditions of the various actions, we define

$$\mathtt{cover}(B) \equiv [U|\mathtt{move}(A,B)] \cup [U|\mathtt{move}(C,B)] \quad (1)$$

the set of all plans that *might* clobber B's being clear, and similarly for $\mathtt{cover}(A)$ and $\mathtt{cover}(C)$. We can now apply Lemma 2.4 to conclude that since B is clear in the initial situation, it will remain clear after executing at *least* the plans in the set

$$\mathtt{clear}(B) \equiv \pi([\,], \mathtt{cover}(B)) \quad (2)$$

since we have perhaps overestimated the set of clobberers in (1). The above set is the set of all action sequences that do not contain either $\mathtt{move}(A,B)$ or $\mathtt{move}(C,B)$, and B will be clear after any such sequence is executed, since it is clear initially.

In a similar way, C is clear for at least the plans in the set $\mathtt{clear}(C) \equiv \pi([\,], \mathtt{cover}(C))$ and C is on A for at least

$$\mathtt{on}(C,A) \equiv \pi([\,], U|\mathtt{move}(C,B) \cup U|\mathtt{move}(C,\mathtt{table}))$$

It follows that A will be made clear by the plans in the set

$$[\mathtt{clear}(C) \cap \mathtt{on}(C,A)]|\mathtt{move}(C,\mathtt{table})$$

and therefore that A will be clear for plans in the set

$$\mathtt{clear}(A) \equiv \pi([\mathtt{clear}(C) \cap \mathtt{on}(C,A)]|\mathtt{move}(C,\mathtt{table}), \mathtt{cover}(A))$$

If C is clear and on A, A will be clear in a persistent way after C is moved to the table. Covering A (the second argument to π above) defeats the persistence.

We can now conclude that A will be on B for at least the plans in the set

$$\mathtt{on}(A,B) \equiv \pi([\mathtt{clear}(A) \cap \mathtt{clear}(B)]|\mathtt{move}(A,B), [U|\mathtt{move}(A,C)] \cup [U|\mathtt{move}(A,\mathtt{table})])$$

In a similar way, B will be on C for at least the plans

$$\mathtt{on}(B,C) \equiv \pi([\mathtt{clear}(B) \cap \mathtt{clear}(C)]|\mathtt{move}(B,C), [U|\mathtt{move}(B,A)] \cup [U|\mathtt{move}(B,\mathtt{table})])$$

The overall goal will be achieved by plans in the set

$$\mathtt{on}(A,B) \cap \mathtt{on}(B,C) \tag{3}$$

Consider now the successful plan

$$\langle \mathtt{move}(C,\mathtt{table}), \mathtt{move}(B,C), \mathtt{move}(A,B) \rangle \tag{4}$$

This plan is in $\mathtt{on}(A,B)$ because the plan itself is in $[\mathtt{clear}(A) \cap \mathtt{clear}(B)]|\mathtt{move}(A,B)$. To see this, we must show that

$$\langle \mathtt{move}(C,\mathtt{table}), \mathtt{move}(B,C) \rangle \tag{5}$$

is in $\mathtt{clear}(A) \cap \mathtt{clear}(B)$. To see *this*, note that no subsequence of (5) is in $\mathtt{cover}(A)$ or $\mathtt{cover}(B)$. As a result, we can show that (5) is in $\mathtt{clear}(A)$ by showing that some subsequence of (5) is in

$$[\mathtt{clear}(C) \cap \mathtt{on}(C,A)]|\mathtt{move}(C,\mathtt{table})$$

In a similar way, we can show that (5) is in $\mathtt{clear}(B)$ by showing that some subsequence of (5) is in $[\,]$.

The second of these is obvious. The first follows because $[\,]$ is in $\mathtt{clear}(C) \cap \mathtt{on}(C,A)$. As a result, we can conclude that A and B are clear after (5), so that A is indeed on B after executing (4). It is not hard to show that B is on C as well, so that (4) does indeed succeed in solving our problem.

What about the failing plan

$$\langle \mathtt{move}(C,\mathtt{table}), \mathtt{move}(A,B), \mathtt{move}(B,C) \rangle \tag{6}$$

where A is moved onto B too early? In this case, we can show that (6) is not known to be in $\mathtt{on}(B,C)$. Specifically, it has no subsequence known to be in

$$[\mathtt{clear}(B) \cap \mathtt{clear}(C)]|\mathtt{move}(B,C)$$

because $\langle \mathtt{move}(C,\mathtt{table}), \mathtt{move}(A,B) \rangle$ is not known to be in $\mathtt{clear}(B) \cap \mathtt{clear}(C)$. Still more specifically, $\langle \mathtt{move}(C,\mathtt{table}), \mathtt{move}(A,B) \rangle$ is not in $\mathtt{clear}(B)$ because the sequence itself is in $\mathtt{cover}(B)$.

3 COMPLEXITY RESULTS

The analysis of the previous section shows that the successful plan

$$\langle \mathtt{move}(C,\mathtt{table}), \mathtt{move}(B,C), \mathtt{move}(A,B) \rangle$$

is a member of (3) while the failing one

$$\langle \mathtt{move}(C,\mathtt{table}), \mathtt{move}(A,B), \mathtt{move}(B,C) \rangle$$

is not. In some sense, this means that membership in (3) resolves the question of whether a specific linear plan satisfies the ordering constraints implicit in the original problem. Our main goal in this section will be to show the following:

Proposition 3.1 *Let $P \in \mathcal{L}$ be a set of plans. Then showing that $P \neq \emptyset$ is NP-complete.*

The proof involves two stages. First, we must show that the problem in question is in NP by demonstrating that it is possible to validate in polynomial time a witness that $P \neq \emptyset$. Second, we need to show that the problem is NP-hard by embedding some known NP-hard problem in it. For the first part, we have:

Definition 3.2 *For any $P \in \mathcal{L}$, we define the* length *of P to be the length of the syntactic expression P itself. The length of P will be denoted $|P|$.*

Lemma 3.3 *Let $P \in \mathcal{L}$ be a nonempty set of plans. Then there is a plan $\langle a_1, \ldots, a_n \rangle \in P$ with $n \leq 1 + |P|$.*

Lemma 3.4 *Let $P \in \mathcal{L}$ be a set of plans, and p a specific linear plan. Then the question of whether $p \in P$ can be resolved in time polynomial in the lengths of p and P.*

Lemma 3.4 tells us that we can validate a witness that $P \neq \emptyset$ in time polynomial in both the size of P and the length of the witness; Lemma 3.3 guarantees that the witness itself is of polynomial size in P.

Corollary 3.5 *Let $P \in \mathcal{L}$ be a set of plans. Showing that $P \neq \emptyset$ is in NP.* ■

To show that the problem is NP-hard, consider the known NP-complete problem of satisfiability. In this problem, we have a set of clauses such as $x_1 \vee \neg x_2 \vee x_3$ and need to find an assignment of true or false to each of the x_i's such that every clause is satisfied. To achieve the embedding, we will view the x_i's as actions as well, taking x_i to mean that x_i is an action in some hypothetical plan p. Now the clauses can all be satisfied if and only if a plan can be found meeting the given conditions.

Definition 3.6 *Let X be a set of atomic variables. Now if $x \in X$ is an atom, we define $p(x)$ to be $\pi(U|x, \emptyset)$. For a literal that is the negation of an atom, we take $p(\neg x) = \neg p(x)$. If $c = \bigvee_i l_i$ is a clause (a disjunction of literals), we take $p(c) = \bigcup_i p(l_i)$, and if $C = \bigwedge_i c_i$ is a theory (a conjunction of clauses), we take $p(C) = \bigcap_i p(c_i)$.*

Lemma 3.7 *A theory C is satisfiable if and only if $p(C) \neq \emptyset$.*

Corollary 3.8 *Let $P \in \mathcal{L}$ be a set of plans. Showing that $P \neq \emptyset$ is NP-hard.* ■

Proposition 3.1 now follows.

4 PLANNING

In this section, we present an algorithm that works from a domain description to compute the set of plans that achieve a particular goal. We will assume that the domain is described in terms of STRIPS-like add, delete, and precondition lists, but this assumption is not central to our methods. We will also assume the existence of an "initializing" action i that sets up the initial state. This action has no preconditions, adds all fluents true in the initial state, and deletes all others.

Let us begin with some fairly high-level remarks. In any realistic situation, it is unlikely that we be able to compute exactly the set of plans achieving a goal g. The axiomatization of the domain can be expected to include information implying the existence of a wide range of exceptions to any specific plan or plan schema that achieves g. These exceptions will themselves have exceptions, plans that achieve g even though they belong to "schema" that are expected not to, and so on. If we denote by $L(g)$ the set of plans that achieve g, we see that it will in general be impractical to attempt to compute $L(g)$ exactly.

Instead, we will give a procedure that computes a set of increasingly fine approximations to $L(g)$. Somewhat more specifically, we will produce sequences of plans

$$L_0^+(g) \supseteq L_1^+(g) \supseteq \cdots$$

and

$$L_0^-(g) \subseteq L_1^-(g) \subseteq \cdots$$

such that

$$L_0^-(g) \subseteq L_1^-(g) \subseteq \ldots \subseteq L(g) \subseteq \ldots \subseteq L_1^+(g) \subseteq L_0^+(g)$$

In other words, $L(g)$ is always bracketed by the L^- and L^+ approximations. Since $L(g) \supseteq L_k^-(g)$ for any k, we can stop and return *some* successful plan as soon as $L_k^-(g) \neq \emptyset$. (Note that the check that $L_k^-(g) \neq \emptyset$ is NP-complete.) Alternatively, we might stop and return when L^+ and L^- are "close" in some sense, as is suggested in the work on approximate planning [9].

My interest here, however, is simply in constructing the approximations L^+ and L^- themselves. To that end, we will think of a node in the planning search tree as corresponding either to a goal or subgoal (a **goal** node) or to a specific action that we intend to take (an **action** node). In addition, we will label nodes with a *parity* in the sense that "positive" nodes support plans to achieve the goal and "negative" nodes correspond to exceptions to those plans or other difficulties. A node can also be *expanded* or *unexpanded*.

Definition 4.1 *A* planning tree *is a singly rooted, directed acyclic graph where no node has paths back to the root of both even and odd length. Nodes at even depths (including the root) will be called* goal *nodes and denoted $(g, \pm)$; nodes at odd depths will be called* action *nodes and denoted $(a, \pm)$. Each fringe node in the graph can be* expanded *or* unexpanded.

If T is a planning tree and n is an unexpanded fringe node in T, the result of expanding n in T *is the planning tree obtained by labeling n as expanded and adding as children of n the following nodes:*

1. If $n = (g, \pm)$ is a goal node, the children are action nodes $(a, \pm)$ for each action that adds g and action nodes $(a, \mp)$ for each action that deletes g.

2. If $n = (a, \pm)$ is an action node, the children are goal nodes $(g, \pm)$ for each precondition g of a.

A legal tree *for a goal g is any tree that can be obtained by successively expanding the tree containing the single node $(g, +)$.*

Of course, it is not sufficient to simply build the trees; we also need to label the nodes with the plans that achieve them. To this end, we define:

Definition 4.2 *In a* conservative *labeling, every unexpanded node in a planning tree with positive parity is labeled $\emptyset$ and every unexpanded node with negative parity is labeled U. In a* liberal *labeling, every unexpanded node with positive parity is labeled U and every unexpanded node with negative parity is labeled $\emptyset$.*

The label for any expanded action node (a, p) is

$$\bigcap_i l(c_i)|a \tag{7}$$

where the c_i are the children of (a, p) and the labels $l(c_i)$ are the labels attached to those children. Any expanded goal node (g, p) is labeled with

$$\pi\left(\bigcup_i l(c_{i,p}), \bigcup_i l(c_{i,\neg p})\right)$$

where the $c_{i,p}$ are the children of the same parity as (g, p) and the $c_{i,\neg p}$ are the children of opposite parity.

The conservative *(respectively* liberal*) labeling of a tree T is the label assigned to the root node by a conservative (respectively liberal) labeling of T. These labelings are denoted $L^-(T)$ and $L^+(T)$ respectively.*

Theorem 4.3 *Let T_0 be a planning tree, and T_1 an expansion of it. Suppose that T_0 is labeled L_0^+ in a liberal labeling and L_0^- in a conservative one, and that T_1 is labeled L_1^+ and L_1^-. Then*

$$L_0^- \subseteq L_1^- \subseteq L(g) \subseteq L_1^+ \subseteq L_0^+$$

where g is the goal at the root of T_0.

The above theorem in isolation tells us little; it would be satisfied if $L_k^- = \emptyset$ and $L_k^+ = U$ for all k! We still need to know that the planning algorithm we have proposed actually converges on the correct answer:

Theorem 4.4 *Let g be a goal, and T_0 the planning tree consisting of the single goal node $(g, +)$. Now if p is a plan that achieves g, there is some expansion T of T_0 such that $p \in L^-(T)$. Conversely, if p fails to achieve g, there is an expansion such that $p \notin L^+(T)$.*

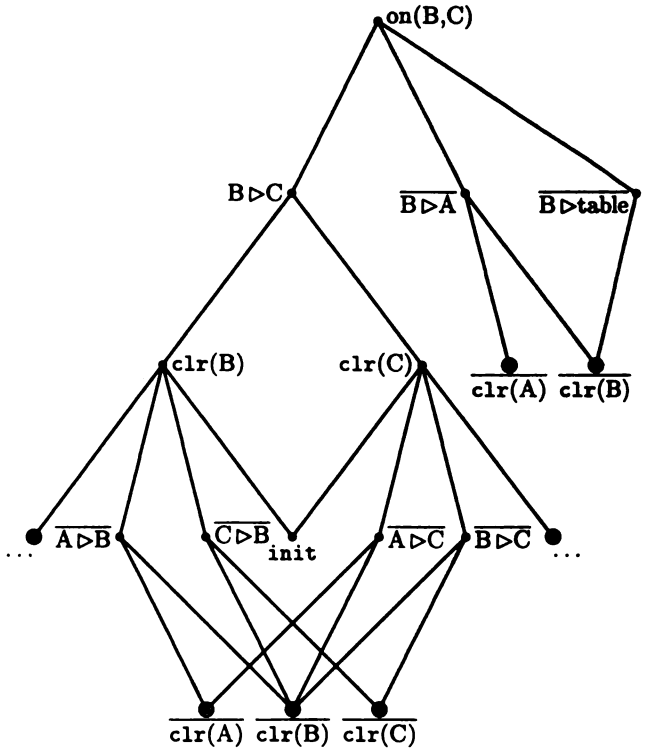


Figure 2: Planning to get B on C

5 COMPUTATION

The results of the previous section sanction the following planning algorithm:

Procedure 5.1 To find a plan for a goal g:

```
T := the planning tree consisting of the single
     node (g, +)
while L⁻(T) = ∅
     if T contains an unexpanded node, expand it
     else fail
return L⁻(T)
```

Theorem 5.2 *If Procedure 5.1 returns a set of plans P in response to a goal g, every plan in P achieves g. In addition, if the nodes are expanded in order of increasing depth, the procedure will always terminate for any achievable goal g.* ■

Procedure 5.1 can be modified to check at each step whether $L^+(T) = \emptyset$. Should this ever happen, it is possible for the planner to report failure even if the tree has not yet been fully expanded.

As an example of the procedure at work, consider the simple goal of getting B onto C in the Sussman anomaly. A partial expansion of the planning tree for this problem appears in Figure 2. The nodes in the

figure are labeled as follows:

1. Action nodes are labeled with the action in question; the action of moving A to B, for example, is denoted $A \triangleright B$. The initializing action is denoted `init`.

2. Goal nodes are labeled with the fluent that the node is trying to achieve. A node labeled `clr`(B), for example, is trying to get B clear.

3. Nodes of negative parity appear with an overline. Nodes of positive parity are not so marked.

4. Unexpanded nodes are marked with a bullseye.

The reasoning behind the arcs in the graph is as follows:

1. The root node is labeled with the goal on(B, C).

2. The root can be achieved by moving B to C. It can be clobbered by moving B to A or to the table. These three children are added at depth 1, with the nodes corresponding to the clobbering actions having negative parity.

3. Moving B to C requires that B and C be clear; moving B to A (a clobbering action) requires that B and A be clear. Since the two nodes at depth two with goal `clear`(B) have opposing parities, they cannot be combined in the planning tree. The two appearances of `clear`(B) under the clobbering actions *are* combined.

4. B is made clear by the initializing action, and also potentially by a variety of other actions that have been lumped into a single node labeled ... in the figure. (Moving A off the top of B will make B clear, and so on.) B being clear will be clobbered by moving either A or C onto B. The goal node labeled `clr`(C) is expanded similarly.

5. The negative parity action node $A \triangleright B$ at depth 3 has as children negative parity goal nodes trying to get A and B clear. These nodes could be identified with analogous nodes at depth 2, but haven't been in the interests of maintaining the clarity of the figure.

What about the labels assigned to the nodes? Assuming that we are interested in a conservative labeling of the tree, we begin by assigning either $\emptyset$ or U to the unexpanded nodes depending on their parity. The fringe nodes at the bottom of the tree (depth 4) and those at depth 2 are all of negative parity and are labeled U, while the unexpanded ... nodes at depth 3 are labeled $\emptyset$.

We can now compute the labels for the nodes at depth 3. These labels include expressions of the form $U|$`init`, indicating that the initial situation needs to be established before the plans involved can be effective. We abbreviate $U|$`init` to [ ] to get the tree in Figure 3. In the figure, we have dropped nodes labeled $\emptyset$ that cannot contribute to the values of nodes at depth 2, and have abbreviated $U|$`move`(A, B) (for example) to $U|$A$\triangleright$B.

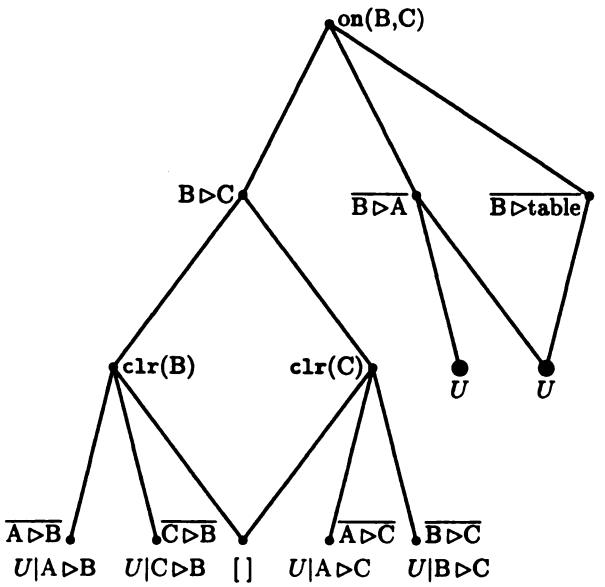


Figure 3: The conservative labeling, continued

Consider now the node at depth 2 in Figure 3 dealing with the goal `clear`(B). The label to be assigned to this node is

$$\pi([\,], U|\texttt{move}(A, B) \cup U|\texttt{move}(C, B))$$

which is exactly the set of plans that we denoted `clear`(B) in (2) of Section 2.

Continuing, the label assigned to the action node $B \triangleright C$ in the figure is seen to be

$$[\texttt{clear}(B) \cap \texttt{clear}(C)]|\texttt{move}(B, C) \qquad (8)$$

and the labels assigned to the two action nodes of negative parity are $U|$`move`(B, A) and $[U|$`move`$(B,$ `table`$)$ respectively. We can finally label the root node as

$$\pi([\texttt{clear}(B) \cap \texttt{clear}(C)]|\texttt{move}(B, C), \\ [U|\texttt{move}(B, A)] \cup [U|\texttt{move}(B, \texttt{table})]) \qquad (9)$$

This plan set is nonempty (the sequence $\langle$`move`$(B, C)\rangle$ is an element of it), and the planner can return at this point.

Note that the planner cannot return until the two negative parity nodes at depth 1 have been expanded. If these were not expanded, (9) would be replaced with

$$\pi([\texttt{clear}(B) \cap \texttt{clear}(C)]|\texttt{move}(B, C), U)$$

which is empty.

After the negative parity nodes at depth two are expanded, we see that specific actions are needed to clobber the goal on(B, C); if these actions are not taken, the goal is achieved. The plan set (9) is nonempty because it is indeed possible to instantiate the plan without clobbering the overall goal.

Note also that although we chose to fully expand the planning tree in this example, there was no intrinsic need for us to do so. If we had identified only some of the actions that might achieve or clobber a particular goal, we would still have been able to construct plan sets in a similar fashion, although the bounds we obtained on the set of plans that actually *do* achieve our goal would necessarily have been somewhat looser.

If we had been working with some other (and presumably more sophisticated) semantics of action, our overall approach would have been unchanged. It would still have been possible for us to construct a tree describing the actions that might achieve or clobber the various goals that were of interest to us, and we would be able to employ quite similar techniques to construct and to then evaluate a sentence in the planning language.

Finally, we remarked in the introduction that unlike conventional planners (POCL or otherwise), Procedure 5.1 plans for subgoals separately. What this means is that once a plan is found for a subgoal anywhere in the tree that is "successful" in that it can be incorporated into a successful plan for the goal at the root, the node corresponding to the subgoal need not be revisited.

Formalizing this is clearest if we assume that nodes with identical parities and labels have not been identified (a conventional planner would be incapable of identifying them in any event). As a preliminary, we make the following definition:

Definition 5.3 *Suppose that T_1 and T_2 are two planning trees with identical roots, and that both are trees as opposed to directed acyclic graphs. We will say that T_1 and T_2* match *at a node n appearing in each of them if (1) the path from n back to the root is the same in the two trees, and (2) the expansions of the two trees under n are identical.*

Proposition 5.4 *Suppose that T is a partially expanded planning tree for a goal g, and that n is a positive node in T. Now if there is any planning tree T' for g that matches T at n and for which $L^-(T') \neq \emptyset$, there is an expansion T_e of T with $L^-(T_e) \neq \emptyset$ such that T_e and T match at n.*

In other words, once we have found an expansion of a node n that participates in any solution of the original planning problem (as witnessed by the tree T'), we need never examine a node below n again.

6 IMPLEMENTATION

We have built a prototype implementation of our ideas that conforms to the theoretical description that we have presented. The planner is called COPS (COmbined Planner and Scheduler).

COPS accepts as input a goal and a set of action descriptors. Each descriptor includes the name of the action, its preconditions, add and delete lists, and a list of resources used by the action. COPS begins by building a planning tree using the mechanisms we have described, ignoring the resource information as it does so. The user can have the system construct the tree in its entirety, or can control the node expansions so that only the portion of interest is built.

Either after the planning tree is complete or while it is being constructed, the user can select any node in the tree for either liberal or conservative evaluation. A sentence in the planning language is then constructed and translated into an equivalent sentence in a finite, sorted first-order logic. This translation incorporates additional constraints that handle potential conflicts due to resource over-utilization by the various actions.

The first-order theory is converted into a (once again equivalent) predicate theory and passed to a predicate logic prover to check for satisfiability. The implementation uses a modification of TABLEAU [3] that incorporates procedural reasoning capabilities [11]. While the original sentence in the planning language directly captures only some of the ordering constraints on the actions involved (and therefore might best be thought of as partial-order), the theory in predicate logic describes conditions that linear plans need to satisfy in order to achieve the overall goals.

The performance of the system is as one would expect. Interactions corresponding to single actions establishing or disestablishing multiple subgoals correspond to entanglements in the planning tree much like those present in Figure 2; these entanglements are eventually resolved by the predicate prover. Interactions that are a consequence of resource contentions are handled by the predicate prover in isolation, with the structure of the planning tree remaining simple in this case.

In the future, we will work to extend COPS in a variety of ways. First, we will exploit the similarity between the planner's existing architecture and the

theorem proving system MVL [7] so that actions can be described using declarative methods instead of a STRIPS-like representation. This will also allow us to treat actions with variables in their description; the current implementation is limited to the ground case.

We will also work to apply COPS to a variety of different domains, and to evaluate its scaling behavior in each. In cases where subgoal interactions correspond to resource contentions, we expect the system's scaling properties to match those of the underlying predicate prover/scheduler; in cases where the interactions correspond to multiple uses of single actions, COPS will doubtless suffer from combinatorial difficulties similar to those encountered by conventional methods. We expect these difficulties to be mediated by the facts that some of the complexity will be deferred to the first-order prover and COPS can recognize repeated subgoals independent of the contexts in which they appear.

7 CONCLUDING REMARKS

The COPS algorithm is only a beginning; there is, for example, no guidance with regard to the selection of the planning node to be expanded next. In this section, we draw connections between our ideas and others that may shed light on this or other issues.

7.1 EXISTING PLANNING WORK

We begin by briefly comparing our ideas and two existing planners, Joslin's DESCARTES [12] and Etzioni's STATIC [5, 6]. DESCARTES is in the POCL class, while STATIC is not.

The DESCARTES planner works to identify the constraint-satisfaction problems corresponding to action scheduling and to then schedule the actions using a dedicated constraint-satisfaction or scheduling algorithm of some sort [12]; by taking this approach, DESCARTES attempts to use the search facilities of the scheduling engine to minimize the search undertaken by the planner per se. This idea is similar to ours, although DESCARTES is capable of making tentative and unforced commitments when it appears that the associated constraint-satisfaction problem has become underconstrained. These "early commitments" appear to have no counterpart in our method.

In the absence of early commitments, the behavior of the two systems is strikingly similar. Both use plans for achieving single subgoals in a variety of contexts, and both suffer from the potential difficulty that problems repeatedly passed to the scheduler share many features with their predecessors. In DESCARTES's case, it would be useful if the results of prior scheduling searches could inform subsequent ones; in our case, the determination of whether $L^-(T) = \emptyset$ can be expected to be similar from one expansion of the planning tree to the next and it will clearly be important to exploit these similarities computationally.

The other connection we would like to draw is between our work and other attempts to combine identical subgoals in planning. At an intuitive level, the development of macro operators [13] is an attempt in this direction. By identifying specific sequences of actions that achieve common goals, the cost of planning can be reduced. Our approach realizes similar advantages if a single sequence of actions can be used to achieve a repeated goal multiple times and can also deal with situations where differing sequences of actions are needed. In this latter case, however, there is some computational cost to our methods, since both sequences will need to be evaluated in the check to see whether $L^-(g) = \emptyset$.

Another author who has attempted to identify recurring subgoals is Etzioni [5]. Etzioni's "problem space graphs" (PSG's) are strikingly similar to our planning trees, consisting as they do of "alternating sequence[s] of subgoals and operators" [5, page 262].[1] The PSG's are not used to solve any planning problem specifically, but instead capture the dependencies between goals and subgoals that are part of the overall problem formulation.

Etzioni describes as "fortuitous" recursions that are of a structural form that can be identified by an analysis of the PSG for the domain in question, and he shows that the blocks world is indeed fortuitous. Unfortunately, no criteria are presented whereby fortuitous recursions can be identified analytically.

We can do a bit better than this. If a node in a planning tree is identical to one of its ancestors, we can denote the label assigned to the node by $L(n)$ and then produce a fixed-point equation that $L(n)$ needs to satisfy. The actual value for $L(n)$ will be the smallest set of plans satisfying the equation. In many cases, it may be possible to show that the label for the ancestor node is in fact independent of the label for the child, in which case we will have shown that the recursion is fortuitous and the child can be pruned.

[1] Smith and Peot's *operator graphs* [15] are also similar, as Dan Weld has pointed out to me. The principal differences between PSG/operator graphs and our approach are that (1) PSG's (but not operator graphs) are constructed statically from the domain description, and not in response to a particular problem instance, and (2) PSG's and operator graphs both include information about goal establish-

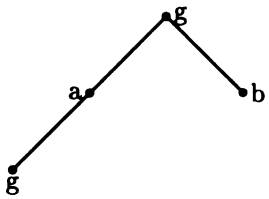


Figure 4: A fortuitous recursion

An example of this phenomenon appears in Figure 4. The goal g has no clobberers and can be achieved by either a or b. The preconditions for a are g, while b has no preconditions.

Suppose that we denote the eventual label for g by L. Now the label for a is $L|a$, while the label for b is $U|b$. The label for g at the root of the tree is therefore

$$\pi(L|a \cup U|b, \emptyset) = L \tag{10}$$

and it is not hard to see that the least solution to (10) is $L = \pi(U|b, \emptyset)$; g is achieved if we ever execute the action b, but in no other cases. In this case, we see that the repeated subgoal g can be pruned without affecting the label of the original. In fact, we can show:

Proposition 7.1 *Any subgoal with no clobberers in its subtree is fortuitous.*

7.2 BILATTICES AND MODALITY

The planning tree structure that we have introduced is similar in many ways to existing work on truth-functional interpretations of modality [8, 10]. Underlying this interpretation is the notion that sentences should be labeled not merely as "true" or "false" but instead with a richer set of truth values selected from a *bilattice* [7]. A bilattice is a set equipped with two partial orders, one indicating how true or false a particular sentence is, and the other partial order indicating how complete one's state of knowledge is.

In this setting, it is possible to respond to a modal query like, "I know it is raining outside," by invoking a theorem prover recursively on the embedded sentential argument ("It is raining outside") and then manipulating the truth value returned as a result [8]. One of the interesting features of naturally occurring modalities is that they tend to either preserve or to invert the "truth" partial order. As an example, "true" (t) is more true than "unknown" (u). The modal operator of knowledge (or necessity), applied to t, returns t. Applied to u, it returns f (false). We might write $K(t) = t$ and $K(u) = f$.

ment only, and ignore deletions and disestablishment.

With regard to the truth partial order, $t >_t u$ and $K(t) >_t K(u)$. The "knowledge" partial order is not preserved; $t >_k u$ but $K(t)$ and $K(u)$ are incomparable because neither corresponds to a more complete state of knowledge than the other. It is not clear why natural modalities should be either t-monotonic or t-antimonotonic in this way.

Now consider our planning language. A set of plans P can be thought of as a function ϕ from the set U of all plans into the two-point set $\{t, f\}$. A specific plan p is in P if and only if $\phi(p) = t$. The set of such functions can be interpreted as a bilattice by first embedding the set $\{t, f\}$ in the four element bilattice $F = \{t, f, u, \bot\}$ and then realizing that F^U, the set of functions from U to F, inherits a bilattice structure from the bilattice structure on F. Suppose that we denote this "planning" bilattice by $\mathcal{P}$.

An operator on sets of plans such as $|$ or π is now nothing more than a function that itself operates on functions from U to $\{t, f\}$. In other words, $|$ and π are binary functions from $\mathcal{P} \times \mathcal{P}$ to $\mathcal{P}$. As such, they can be interpreted as modal operators on the planning bilattice. The bilattice-based modal theorem prover [10] is, in fact, little more than a lifting of Procedure 5.1 to a more general setting. Once again, the modalities preserve or invert the t partial order. This is clear for $|$; increasing the set of plans in P or Q increases the set of plans in $P|Q$. The operator π, on the other hand, is monotonic in its first argument and antimonotonic in its second. Perhaps the planning results will shed some light on the bilattice theory here, since we know from results in this paper that the monotonicity and antimonotonicity underlie the convergence properties of the overall algorithms.

7.3 APPROXIMATE PLANNING

Finally, consider the termination criteria in Procedure 5.1: We stop as soon as $L^-(g) \neq \emptyset$, showing that the set of plans $L(g)$ for achieving the goal g is nonempty. There is an optional check to see if $L^+(g) = \emptyset$, in which case $L(g) = \emptyset$ as well and we can terminate and report that the goal is not achievable.

There are other possibilities also. One is to stop whenever the bracketing sets $L^+(T)$ and $L^-(T)$ are in some sense "close" to one another; this is similar to suggestions made in the work on *approximate planning* [9].

The basic idea here is to define conditions under which one set of plans is "small" relative to another. As an example, the set of plans $\pi(P|a, \emptyset)$ will typically be small in the set $\pi(P, \emptyset)$ since the first set involves a commitment to take the action a while the second set

does not.

We can now go on to say that two sets $P \subseteq Q$ are *approximately equal*, writing $P \approx Q$, if their difference $Q - P$ is small relative to Q. Consider, for example, the plan

$$\pi(U|\mathtt{move}(B,C),\emptyset) \tag{11}$$

for getting B onto C in the Sussman anomaly. In order for this plan to fail, one of a handful of specific actions must be taken: Something (A or C) needs to be moved onto B, something needs to be moved onto C, or B needs to be moved away after it is moved onto C successfully. Each of these "exception" plans is small relative to the overall plan schema (11), so (11) itself is approximately equal to the set of plans that actually achieve the goal.

We can modify our basic procedure to return these approximate plans by changing the termination condition from $L^-(T) \neq \emptyset$ to $L^-(T) \approx L^+(T)$. Doing so produces a procedure that implements the approximate planning ideas that have appeared elsewhere, although the complexity of checking whether $P \approx Q$ for $P \subseteq Q$ and $P, Q \in \mathcal{L}$ is not yet known.

All told, the ideas and algorithms that we have described seem to raise as many new questions as they answer. The new questions are, however, quite different from those that have typically been considered by the generative planning community. Perhaps the differences themselves are an indication of progress.

A PROOFS

Lemma 2.6 *U is the set of all linear plans. Negation and intersection have their conventional meanings.*

Proof. To see that $\langle a_1, \ldots, a_n\rangle \in U$ for any plan, note that the longest subsequence of $\langle a_1, \ldots, a_n\rangle$ in $[\,] \cup \emptyset$ is the initial empty subsequence, which is not in $\emptyset$.

For negation, Suppose that $\langle a_1, \ldots, a_n\rangle \in P$. Now it is in $U \cup P$, and also in P, so $\langle a_1, \ldots, a_n\rangle \notin \neg P = \pi(U, P)$. Alternatively, if $\langle a_1, \ldots, a_n\rangle \notin P$, it is in $U \cup P = U$ anyway, so $\langle a_1, \ldots, a_n\rangle \in \neg P$. The result for intersection follows from this. ■

Lemma 3.3 *Let $P \in \mathcal{L}$ be a nonempty set of plans. Then there is a plan $\langle a_1, \ldots, a_n\rangle \in P$ with $n \leq 1 + |P|$.*

Proof. Let $\langle a_1, \ldots, a_k\rangle$ be a linear plan. We will show that there is some subset S of the indices $\{1, \ldots, k\}$ with the size of S at least $k - 1 - |P|$ and such that we can remove any subset of the actions in $\langle a_1, \ldots, a_k\rangle$ whose indices are in S without affecting membership in P. The result follows; given any element of P containing k actions, we can drop at least $k - 1 - |P|$ of them to get an element of P containing at most $1 + |P|$ actions.

The proof is by structural induction on the set of plans P. For any of the base cases $\emptyset$, $[\,]$ or $a \in A$, the result is clear. (At most two actions are needed to resolve the question of membership in P, and $|P| = 1$.)

For the compound cases, consider $P|Q$ first. There is now a set S_p of appearances in P where actions can be dropped, and a set S_q of similar appearances in Q. We take $S = S_p \cap S_q$, so that membership in both P and Q is unaffected. Now the size of S is at least

$$\begin{aligned} |S_p| + |S_q| - k &\geq k - 1 - |P| + k - 1 - |Q| - k \\ &= k - 1 - (|P| + |Q| + 1) \\ &= k - 1 - |(P|Q)| \end{aligned}$$

since the size of $P|Q$ includes an additional count for the concatenation symbol. $\bigcup$ and π can be handled identically. ■

Lemma 3.4 *Let $P \in \mathcal{L}$ be a set of plans, and p a specific linear plan. Then the question of whether $p \in P$ can be resolved in time polynomial in the lengths of p and P.*

Proof. The only subtlety involves the clauses defining $P|Q$ and $\pi(P, Q)$ in Definition 2.3, since these clauses ostensibly involve a search over subsequences of p. We take a dynamic programming approach, investigating all of the subsequences of p simultaneously and working from the component plans of P outward.

Somewhat more specifically, suppose that Q is a set of plans, $p = \langle a_1, \ldots, a_n\rangle$, and for each top level plan Q_i appearing in Q and each $j \leq k \leq n$, we know whether $\langle a_j, \ldots, a_k\rangle \in Q_i$. Now we can determine in time $o(k)$ whether $\langle a_1, \ldots, a_k\rangle \in Q$. If Q is the concatenation of two plans, we consider each point at which the first plan might stop and the second begin. If Q is the result of applying π, we consider each initial subsequence of p starting with the longest (i.e., p itself).

Since there are at most n^2 subsequences of the plan p, it follows that we can determine membership in Q for each of them in time at most n^3 overall. If l is the length of the original set of plans P, it follows that we can determine in time $o(n^3 l)$ whether $\langle a_j, \ldots, a_k\rangle \in P$ for each j and k, and thus whether $p \in P$. ■

Lemma 3.7 *A theory C is satisfiable if and only if $p(C) \neq \emptyset$.*

Proof. It is clear from Definition 3.6 that the result will follow if we can show that for any atom x_j and linear plan $\langle x_1, \ldots, x_n\rangle$, $\langle x_1, \ldots, x_n\rangle \in p(x_j)$ if and only if $x_j \in \{x_i\}$ This is immediate from the definition,

however. ■

Theorem 4.3 *Let T_0 be a planning tree, and T_1 an expansion of it. Suppose that the root of T_0 is labeled L_0^+ in a liberal labeling and L_0^- in a conservative one, and that the root of T_1 is labeled L_1^+ and L_1^-. Then*

$$L_0^- \subseteq L_1^- \subseteq L(g) \subseteq L_1^+ \subseteq L_0^+ \qquad (12)$$

where g is the goal at the root of T_0.

Proof. The proof consists of two parts. In the first, we show that $L_0^- \subseteq L_1^-$; in the second, that $L_0^- \subseteq L(g)$. Since the argument is independent of the actual tree T_0, it will follow that $L_1^- \subseteq L(g)$ as well. The right hand inclusions in (12) follow using dual arguments.

To see that $L_0^- \subseteq L_1^-$, suppose that T is a planning tree, and T' an expansion of it. Suppose also that n is any node in T. If a conservative labeling of T labels n with P while a conservative labeling of T' labels n with P', we claim that $P \subseteq P'$ if the parity of n is positive, and $P \supseteq P'$ if the parity of n is negative.

We show this by induction from the fringe of the tree. For n an unexpanded node in T', the labels are identical in the two cases. If n is the node expanded to produce T' from T, the result follows immediately from the definition of a conservative labeling, since n will be labeled either with U (negative parity) or with $\emptyset$ (positive parity) in T.

For the inductive step, suppose that n is an internal node in the tree. If n is an action node, then the appropriate inequality holds for each of n's children by virtue of the inductive hypothesis, and therefore for n itself because of the definition (7). If n is a goal node, the result is a consequence of Lemma 2.4.

This completes the proof that $L_0^- \subseteq L_1^-$. To show that $L_0^- \subseteq L(g)$, we show the following:

1. The conservative labeling of T labels an action node $(a, +)$ with a plan that is a subset of the set of plans in which a has just been executed successfully.
2. The conservative labeling of T labels an action node $(a, -)$ with a superset of the set of plans in which a has just been executed successfully.
3. The conservative labeling of T labels a goal node $(g, +)$ with a subset of the set of plans in which g holds.
4. The conservative labeling of T labels a goal node $(g, -)$ with a superset of the set of plans in which g holds.

Once again, the proof proceeds by induction from the fringe of the tree; once again, that the above results hold for the fringe itself is obvious.

For an internal action node, the results hold because the set of plans in which a has just been executed successfully is exactly the intersection of the plans for which each of the preconditions of a holds, followed by the execution of the action a itself. For an internal goal node $(g, \pm)$, the set of plans for which g holds can be obtained by propagating the actions that successfully add or delete g. ■

Theorem 4.4 *Let g be a goal, and T_0 the planning tree consisting of the single goal node $(g, +)$. Now if p is a plan that achieves g, there is some expansion T of T_0 such that $p \in L^-(T)$. Conversely, if p fails to achieve g, there is an expansion such that $p \notin L^+(T)$.*

Proof. In fact, we show something stronger. Suppose that the length of p is k. Now we prove that if T is any tree with no unexpanded nodes shallower than depth $2k + 2$, $p \in L^-(T)$ if and only if $p \in L^+(T)$. Since $L^-(T) \subseteq L(g) \subseteq L^+(T)$, the result will then follow.

The proof is by induction on k. If $k = 0$, so that $p = [\,]$, expanding the root node of T will produce an action node requiring *some* action (perhaps the initializing action i) to be taken to establish g. It follows that $[\,]$ will be in either both $L^-(T)$ and $L^+(T)$ or in neither.

For the inductive step, suppose that T has been fully expanded to depth $2k + 2$. Now if n is a node in T of depth 2, we can apply the inductive hypothesis to conclude that L^- and L^+ agree on n for plans of length k or shorter. It follows that they will also agree on all (action) nodes of depth 1 and plans of length $k + 1$ or less, since each of the depth 1 nodes requires the inclusion of a new final action. Since determining whether a plan p is in the set corresponding to the label for the root of the tree involves considering subsequences of p, the result for the root now follows from the result for the children at depth 1. ■

Proposition 5.4 *Suppose that T is a partially expanded planning tree for a goal g, and that n is a positive node in T. Now if there is any planning tree T' for g that matches T at n and for which $L^-(T') \neq \emptyset$, there is an expansion T_e of T with $L^-(T_e) \neq \emptyset$ such that T_e and T match at n.*

Proof. We can take T_e to be the union of T and T'. Since T_e is also an expansion of T', the result follows immediately from Theorem 4.3. ■

Proposition 7.1 *Any subgoal with no clobberers in its subtree is fortuitous.*

Proof. The situation is quite like that of Figure 4. Suppose that we label the repetition of the goal g with $\emptyset$ and that after doing so, conclude that $L(g) = P$. These are the plans that can be used for achieving g without recursion.

Now suppose we label the repeated appearance of g with L. Then the action node a_1 above g will be labeled with some subset of $L|a$. If the goal node above a_1 was previously labeled $Q = \pi(Q^+, \emptyset)$, it will now be labeled with a subset of

$$\begin{aligned} Q' &= \pi(Q^+ \cup L|a_1, \emptyset) = \pi(Q^+, \emptyset) \cup \pi(L|a_1, \emptyset) \\ &= Q \cup \pi(L|a_1, \emptyset) \end{aligned}$$

Continuing up the tree, if the path from the repeated occurrence of g back to the original one involves the actions $a_1, \ldots, a_n$, we see that the eventual label assigned to the root g will be a subset of

$$\pi(L|a_1|a_2|\ldots|a_n, \emptyset) \cup P$$

We therefore need to show that

$$\pi(P|a_1|a_2|\ldots|a_n, \emptyset) \cup P = P$$

or that

$$\pi(P|a_1|a_2|\ldots|a_n, \emptyset) \subseteq P$$

To see this, note that P itself is of the form $\pi(P^+, \emptyset)$ for some P^+, so that if we denote $\langle a_1, \ldots, a_n \rangle$ by a, we are trying to show

$$\pi(\pi(P^+, \emptyset)|a, \emptyset) \subseteq \pi(P^+, \emptyset) \qquad (13)$$

But now let p be any element of the left hand side of (13). Since some initial subsequence of p is in $\pi(P^+, \emptyset)|a$, it follows that some initial subsequence of p is in $\pi(P^+, \emptyset)$, and therefore that some initial subsequence of p is in P^+. Thus p itself is an element of $\pi(P^+, \emptyset)$, and (13) follows as desired. The proof is complete. ■

Acknowledgment

This work has been supported by ARPA/Rome Labs under contracts F30602-91-C-0036 and F30602-93-C-00031. I would like to thank David Etherington, Oren Etzioni, David Joslin, Bart Massey, David McAllester, Dan Weld, and the members of CIRL for discussing these ideas with me.

References

[1] T. Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69:165–204, 1994.

[2] D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333–377, 1987.

[3] J. M. Crawford and L. D. Auton. Experimental results on the crossover point in random 3sat. *Artificial Intelligence*, 81:13–59, 1996.

[4] K. S. Erol, D. S. Nau, and V. Subrahmanian. Complexity, decidability and undecidability results for domain-independent planning. *Artificial Intelligence*, 76:75–88, 1995.

[5] O. Etzioni. Acquiring search-control knowledge via static analysis. *Artificial Intelligence*, 62:255–302, 1993.

[6] O. Etzioni. A structural theory of explanation-based learning. *Artificial Intelligence*, 60:93–140, 1993.

[7] M. L. Ginsberg. Multivalued logics: A uniform approach to reasoning in artificial intelligence. *Computational Intelligence*, 4:265–316, 1988.

[8] M. L. Ginsberg. Bilattices and modal operators. *Journal of Logic and Computation*, 1:41–69, 1990.

[9] M. L. Ginsberg. Approximate planning. *Artificial Intelligence*, 76:89–123, 1995.

[10] M. L. Ginsberg. Modality and interrupts. *J. Automated Reasoning*, 14:43–91, 1995.

[11] A. K. Jonsson and M. L. Ginsberg. Efficient reasoning using procedures. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning*, Boston, MA, 1996.

[12] D. Joslin and M. Pollack. Passive and active decision postponement in plan generation. In *European Workshop on Planning*, 1995.

[13] R. E. Korf. Macro-operators: A weak method for learning. *Artificial Intelligence*, 26:35–77, 1985.

[14] B. Selman. Near-optimal plans, tractability, and reactivity. In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning*, Bonn, Germany, 1994.

[15] D. E. Smith and M. A. Peot. Postponing threats in partial-order planning. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 500–506, 1993.

Moving a Robot: The KR&R Approach at Work

Giuseppe De Giacomo, Luca Iocchi, Daniele Nardi, Riccardo Rosati
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
email: {degiacomo,iocchi,nardi,rosati}@dis.uniroma1.it

Abstract

The paper describes an approach to reasoning about actions and plan generation within the framework of description logics. From an epistemological viewpoint, our approach is based on the formalization of actions given by dynamic logics, but we exploit their correspondence with description logics to turn the formalization into an actual implementation. In particular, we are able to carefully weaken the logical inference process, thus making the reasoning of the robot computationally feasible. From a practical viewpoint, we use a general purpose knowledge representation environment based on description logics, and its associated reasoning tools, in order to plan the actions of the mobile robot "Tino", starting from the knowledge about the environment and the action specification. The robot's reactive capabilities allow it to execute such plans in the real world.

1 INTRODUCTION

We present one attempt to reconcile the theoretical work in knowledge representation with the implementation of real systems. The realm we address is that of mobile robots, which has always been considered central to Artificial Intelligence. Recent work in this field (see for example [Brooks,1986]) has shown that, in order to enable a mobile robot to cope with the uncertainties and dynamics of real environments, some kind of reactive behavior is necessary. However, a mobile robot needs not only the ability to promptly react and adjust its behavior based on the information acquired through its sensors, but also to achieve high-level goals. Therefore, it should also be able to reason about the actions it can perform, find plans that allow it to achieve its goals and check whether the execution of actions leads to the accomplishment of the goals. The integration of reactive and planning capabilities has thus become a focus of the research in mobile robots and planning systems (see for example [Saffiotti *et al.*,1995; Kaelbling and Rosenschein,1995; Simmons,1992; Gat,1992]).

In the present work, we provide a framework for reasoning about actions and discuss its implementation, through a knowledge-based system, on a robot with reactive capabilities. Our approach falls in the research stream of logic-based approaches for reasoning about actions (see [Lesperance *et al.*,1994]), however it has been developed as a balance between theoretical and practical considerations. Specifically, the basis of our proposal for reasoning about actions is provided by Propositional Dynamic Logics (PDLs), following the work of [Rosenschein,1981; De Giacomo and Lenzerini,1995b]. In this setting PDLs formulae denote properties of states, and actions (also called programs) denote state transitions from one state to another. The dynamic system itself is described by means of axioms. Two kinds of axioms are introduced, "static axioms", that describe background knowledge, and "dynamic axioms", that describe how the situation changes when an action is performed. As in the deductive-planning tradition, a plan can be generated by finding a constructive existence proof for the state where the desired goal is satisfied. In a PDL setting a plan consists of a sequence of transitions, which leads to a state satisfying the goal.

The novel and fundamental step towards the implementation has been to rely on the tight correspondence that exists between PDLs and Description Logics (DLs) [Schild,1991; De Giacomo and Lenzerini,1994]. By exploiting this correspondence we have been able both to develop an interesting theoretical framework for reasoning about actions and to obtain an implementation that uses a knowledge representation system based on DLs.

The work on efficient reasoning methods in DLs shows that the typical form of dynamic axioms is problematic wrt efficiency (such axioms are "cyclic" in the DLs terminology). Hence we have reinterpreted dynamic

axioms by means of the so-called procedural rules. By relying on the epistemic interpretation of these rules given in [Donini *et al.*,1994] we have defined a setting which provides both an epistemic representation of dynamic axioms and a weak form of reasoning. In this way, we obtain a computationally feasible and semantically justified approach to deductive planning.

Indeed, there are several studies that propose to use DLs as a basis for the development of planning systems (among them [Artale and Franconi,1994; Borgida,1992; Koehler,1994]). These works extend the DLs language with specific constructs that allow actions to be represented as concepts. The planning system can thus reason about plans, by exploiting subsumption in DLs. Our proposal takes a different perspective, derived from the correspondence with PDLs, where actions are represented as roles, and properties of states as concepts. In our case, plans are generated through a combination of the propagation mechanism for the procedural rules and taxonomic reasoning for checking the static properties of states.

We have built an implementation on top of the mobile robot *Erratic*, equipped with wheels and sonar, which has the capability of integrating action execution and reactive behavior [Konolige,1995]. The knowledge representation system used in the implementation is CLASSIC [Borgida *et al.*,1989], a well-known, general-purpose knowledge representation system based on DLs. One interesting feature of the implementation is that it relies on the reasoning tools provided by such a system, although in this way the formalization is restricted to the subset of theories expressible in the CLASSIC representation language. We named our mobile robot "Tino" and demonstrated it at the 1995 Description Logic Workshop.

The paper is organized as follows. In Section 2, we present the general framework for the representation of dynamic systems we have adopted. In Section 3 we introduce Epistemic DLs, and in Section 4 we address our specific way of representing and reasoning about actions in such a formalism. Finally, in Section 5, we describe the mobile robot "Tino", which includes the implementation in CLASSIC of the planning component and a module for exchanging information between high-level planning and the software implementing the reactive capabilities of the robot.

2 REASONING ABOUT ACTIONS: THE GENERAL FRAMEWORK

In this section we present the general framework for representing dynamic systems on which our work is based. Such a framework is essentially that of PDLs [Rosenschein,1981; De Giacomo and Lenzerini,1995a].

Dynamic systems are typically modeled in terms of state evolutions caused by actions. A *state* represents a situation the system can be in, and is characterized by a set of properties which forms a *complete* description (wrt some logic/language) of the represented situation. *Actions*, which are typically considered deterministic, cause state transitions, making the system evolve from the current state to the next one.

In principle we could represent the *behavior* of a system, i.e. all its possible evolutions, as a *transition graph*, where each node denotes a state, and is labeled with the properties that characterize the state, and each arc denotes a state transition, and is labeled with the action that causes the transition. Note, however, that *complete knowledge* of the behavior of the system is required to build its transition graph, while in general one has only partial knowledge of such behavior. In deductive planning this knowledge is phrased in *axioms* of some logic (e.g. PDLs [Rosenschein,1981] or Situation Calculus [Reiter,1993]). These axioms select a subset of all possible transition graphs. All the selected graphs are similar, since they all satisfy the same axioms, but yet different wrt those properties not imposed by the axioms. Hence one has to concentrate on those properties that are true in all the selected graphs, i.e. those properties that are *logically implied* by the axioms.

Following [Rosenschein,1981] two kinds of axioms are distinguished:

- *Static axioms*, which are used for representing background knowledge that is invariant with respect to the execution of actions. In other words, static axioms hold in any state and do not depend on actions.
- *Dynamic axioms*, which are introduced to represent the changes actions bring about, and have the form[1]:
$$C \Rightarrow \langle R \rangle \mathtt{tt} \wedge [R]D$$
where R is an action, C represents the *preconditions* that must hold in a state, in order for the action R to be executable; D denotes the *postconditions* that are true in the state resulting from the execution of R in a state where preconditions C hold. Multiple axioms per action are allowed.

In deductive planning one is typically interested in answering the following question: "Is there a sequence of actions that, starting from an initial state, leads to a state where a given property (the goal) holds?". Under the assumption of deterministic actions, this is captured by the following logical implication (here we phrase it in PDLs):

$$\Gamma \models S \Rightarrow \langle \alpha^* \rangle G \qquad (1)$$

where: (i) Γ is the set of both static and dynamic axioms representing the (partial) knowledge about the

[1] Actually in [Rosenschein,1981] the form of the dynamic axioms is $C \Rightarrow [R]D$, and for each action R the axiom $\langle R \rangle \mathtt{tt}$ is assumed to be valid.

DLs		PDLs	
atomic concept	A	atomic proposition	A
top	$\top$	true	$\mathtt{tt}$
bottom	$\bot$	false	$\mathtt{ff}$
conjunction	$C \sqcap D$	conjunction	$C \wedge D$
disjunction	$C \sqcup D$	disjunction	$C \vee D$
negation	$\neg C$	negation	$\neg C$
existential quantification	$\exists R.C$	diamond ("some runs ...")	$\langle R \rangle C$
universal quantification	$\forall R.C$	box ("all runs ...")	$[R]C$
inclusion assertion	$C \sqsubseteq D$	valid implication (axiom)	$C \Rightarrow D$
instance assertion	$C(\alpha) \mid R(\alpha_1, \alpha_2)$	—	—

Figure 1: Correspondence between DLs and PDLs.

system; (ii) S is a formula representing the (partial) knowledge about the initial situation (state); (iii) G is a formula representing the goal, which is, in fact, a (partial) description of the final state one wants to reach; (iv) $\langle \alpha^* \rangle G$ (where $\langle \alpha^* \rangle G$ stands for any formula of the form $\langle R_1 \rangle \langle R_2 \rangle \ldots \langle R_n \rangle G$ with $n \geq 0$ and R_i any action) expresses the existence of a finite sequence of actions leading to a state where G is satisfied.

From a *constructive proof* of the above logical implication one can extract an actual sequence of actions (a plan) that leads to the goal.

Observe that in this setting one may have a very sparse knowledge about the system – say a few laws (axioms) one knows the system obeys – and yet be able to make several non-trivial inferences. Unfortunately, this generality incurs a high computational cost (typically PDLs are EXPTIME-complete [Kozen and Tiuryn,1990]).

Therefore, we make use of the correspondence between PDLs and Description Logics (DLs) to take advantage of the extensive studies of the computational aspects of reasoning in DLs, and to exploit specific techniques developed in DLs to lower the cost of reasoning. We present our proposal using the notation of DLs, in order to make it easier to relate our proposal both to previous research in DLs, whose results are exploited here, and to the actual implementation in CLASSIC.

3 EPISTEMIC DESCRIPTION LOGICS

In this section we introduce description logics with an epistemic operator, which constitute the technical background of our proposal. We focus on a well-known DL, $\mathcal{ALC}$, and its epistemic extension, $\mathcal{ALCK}$, obtained by adding a modal operator interpreted in terms of minimal knowledge as in [Donini *et al.*,1992; 1994; 1995].

In Fig. 1 we present the constructs of the DL $\mathcal{ALC}$ and the correspondence with PDLs. Such correspondence, first pointed in [Schild,1991], is based on the similarity between the interpretation structures of the two kinds of logics. At the extensional level, states in PDLs correspond to individuals (members of the domain of interpretation) in DLs, whereas state transitions correspond to links between two individuals. At the intensional level, propositions correspond to concepts, and actions correspond to roles. The correspondence is realized through a (one-to-one and onto) mapping from PDLs formulae to DLs concepts, and from PDLs actions to DLs roles. For a detailed presentation of such a mapping and more generally of the correspondence we refer to [Schild,1991; De Giacomo and Lenzerini,1994]. For our purposes it suffices to consider DLs concepts and roles as syntactic variants of PDLs formulae and actions respectively.

$$
\begin{aligned}
\top^{\mathcal{I}} &= \Delta \\
\bot^{\mathcal{I}} &= \emptyset \\
A^{\mathcal{I}} &\subseteq \Delta \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} &= \Delta \setminus C^{\mathcal{I}} \\
\forall R.C^{\mathcal{I}} &= \{d_1 \in \Delta \mid \\
&\qquad \forall d_2 . (d_1, d_2) \in R^{\mathcal{I}} \Rightarrow d_2 \in C^{\mathcal{I}}\} \\
\exists R.C^{\mathcal{I}} &= \{d_1 \in \Delta \mid \\
&\qquad \exists d_2 . (d_1, d_2) \in R^{\mathcal{I}} \wedge d_2 \in C^{\mathcal{I}}\}
\end{aligned}
$$

Figure 2: Semantics of $\mathcal{ALC}$

In the following we shall refer to an *$\mathcal{ALC}$-interpretation* $\mathcal{I}$ as a function mapping each concept expression into a subset of some abstract interpretation domain Δ and each role expression into a subset of $\Delta \times \Delta$, such that the equations of Fig. 2 are satisfied.

The basic reasoning service for DLs is subsumption: C is subsumed by D if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every $\mathcal{I}$. In

$$
\begin{aligned}
\top^{\mathcal{I},\mathcal{W}} &= \Delta \\
\bot^{\mathcal{I},\mathcal{W}} &= \emptyset \\
(\mathbf{K}C)^{\mathcal{I},\mathcal{W}} &= \bigcap_{\mathcal{J}\in\mathcal{W}} (C^{\mathcal{J},\mathcal{W}}) \\
(C \sqcap D)^{\mathcal{I},\mathcal{W}} &= C^{\mathcal{I},\mathcal{W}} \cap D^{\mathcal{I},\mathcal{W}} \\
(C \sqcup D)^{\mathcal{I},\mathcal{W}} &= C^{\mathcal{I},\mathcal{W}} \cup D^{\mathcal{I},\mathcal{W}} \\
(\neg C)^{\mathcal{I},\mathcal{W}} &= \Delta \setminus C^{\mathcal{I},\mathcal{W}} \\
(\forall R.C)^{\mathcal{I},\mathcal{W}} &= \{d_1 \in \Delta \mid \\
&\quad \forall d_2.(d_1,d_2) \in R^{\mathcal{I},\mathcal{W}} \Rightarrow d_2 \in C^{\mathcal{I},\mathcal{W}}\} \\
(\exists R.C)^{\mathcal{I},\mathcal{W}} &= \{d_1 \in \Delta \mid \\
&\quad \exists d_2.(d_1,d_2) \in R^{\mathcal{I},\mathcal{W}} \wedge d_2 \in C^{\mathcal{I},\mathcal{W}}\} \\
(\mathbf{K}P)^{\mathcal{I},\mathcal{W}} &= \bigcap_{\mathcal{J}\in\mathcal{W}} (P^{\mathcal{J},\mathcal{W}}).
\end{aligned}
$$

Figure 3: Semantics of $\mathcal{ALCK}$

other words subsumption allows to establish a hierarchy among concept descriptions, which can be used to reason on a specific problem instance. The complexity of subsumption in DLs has been carefully studied for different logics obtained by admitting different sets of constructs.

$\mathcal{ALCK}$ is an extension of $\mathcal{ALC}$ with an epistemic operator interpreted as knowledge. More precisely the $\mathcal{ALCK}$ abstract syntax is as follows (C, D denote concepts, R denotes a role, A denotes a primitive concept, and P a primitive role):

$$
\begin{aligned}
C, D \quad &::= \quad A \mid \top \mid \bot \mid C \sqcap D \mid C \sqcup D \mid \neg C \mid \\
&\qquad \forall R.C \mid \exists R.C \mid \mathbf{K}C \\
R \quad &::= \quad P \mid \mathbf{K}P
\end{aligned}
$$

Non-epistemic concepts and roles are given essentially the standard semantics of DLs, conversely epistemic sentences are interpreted on the class of Kripke structures where worlds are $\mathcal{ALC}$-interpretations, and all worlds are connected to each other, i.e. the accessibility relation among $\mathcal{ALC}$-interpretations is universal.

The semantics is based on the Common Domain Assumption: Every interpretation is defined over the same, fixed, countable-infinite domain of individuals Δ.

An *$\mathcal{ALCK}$-interpretation* is defined as a pair $(\mathcal{I}, \mathcal{W})$ where $\mathcal{W}$ is a set of $\mathcal{ALC}$-interpretations over the domain Δ, and $\mathcal{I}$ is a distinguished interpretation belonging to $\mathcal{W}$ (i.e. $\mathcal{I} \in \mathcal{W}$), such that $A^{\mathcal{I},\mathcal{W}} \subseteq \Delta$, $P^{\mathcal{I},\mathcal{W}} \subseteq \Delta \times \Delta$ and the equations in Fig. 3 are satisfied.

Intuitively, an individual $d \in \Delta$ is an instance of a concept C iff $d \in C^{\mathcal{I},\mathcal{W}}$ in the particular $\mathcal{ALC}$-interpretation $\mathcal{I} \in \mathcal{W}$. An individual $d \in \Delta$ is an instance of a concept $\mathbf{K}C$ (i.e. $d \in (\mathbf{K}C)^{\mathcal{I},\mathcal{W}}$) iff $d \in C^{\mathcal{J},\mathcal{W}}$ for all possible $\mathcal{ALC}$-interpretations $\mathcal{J} \in \mathcal{W}$. In other worlds, an individual is known to be an instance of a concept if it belongs to the concept interpretation of every possible world. Similarly, an individual $d \in \Delta$ is an instance of a concept $\exists \mathbf{K}R.\top$ iff there is an individual $d' \in \Delta$ such that $(d, d') \in R^{\mathcal{J},\mathcal{W}}$ for all possible $\mathcal{J} \in \mathcal{W}$.

DLs are typically used for representing the knowledge about a problem domain by providing mechanisms both for introducing concept definitions and for specifying information about individuals. Accordingly, an *$\mathcal{ALCK}$ knowledge base* Ψ is defined as a pair $\Psi = \langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T}$, called the *TBox*, is a finite set of inclusion statements of the form $C \sqsubseteq D$, with $C, D \in \mathcal{ALCK}$, and $\mathcal{A}$, called the *ABox*, is a finite set of membership assertions of the form $C(a)$ or $R(a, b)$, where $C, R \in \mathcal{ALCK}$ and a, b are *names* of individuals. We assume that different names denote different individuals, hence, we generally do not distinguish between individuals and their names.

The truth of inclusion statement is defined in terms of set inclusion: $C \sqsubseteq D$ is satisfied iff $C^{\mathcal{J},\mathcal{W}} \subseteq D^{\mathcal{J},\mathcal{W}}$. Assertions are interpreted in terms of set membership: $C(a)$ is satisfied iff $a \in C^{\mathcal{J},\mathcal{W}}$ and $R(a, b)$ is satisfied iff $(a, b) \in R^{\mathcal{J},\mathcal{W}}$. A *model* for an $\mathcal{ALCK}$-knowledge base Ψ is a set of $\mathcal{ALC}$-interpretations $\mathcal{W}$ such that for each interpretation $\mathcal{I} \in \mathcal{W}$, every sentence (inclusion or membership assertion) of Ψ is true in the $\mathcal{ALCK}$-interpretation $(\mathcal{I}, \mathcal{W})$.

The final step of the construction consists of defining a preference semantics on universal Kripke structures, which allows one to select only those model where the knowledge is minimal. This is achieved by maximizing in each epistemic model the number of possible worlds (i.e. $\mathcal{ALC}$-interpretations), which can also be explained as maximizing ignorance.

A *preferred model* $\mathcal{W}$ for Ψ is a model for Ψ such that $\mathcal{W}$ is a maximal set of $\mathcal{ALC}$-interpretations, in the sense that for each set $\mathcal{W}'$, if $\mathcal{W} \subset \mathcal{W}'$ then $\mathcal{W}'$ is not a model for Ψ. Ψ is *satisfiable* if there exists a preferred model for Ψ, *unsatisfiable* otherwise. Ψ logically implies an (inclusion or membership) assertion σ, written $\Psi \models \sigma$, if σ is true in every preferred model for Ψ.

Using the epistemic operator, it is possible to formalize in $\mathcal{ALCK}$ several interesting features provided by frame systems, based on DLs [Donini *et al.*,1994]. In particular, here we recall the so-called *procedural rules* (or simply rules).

Procedural rules take the form:

$$C \mapsto D$$

(where C, D are concepts). Roughly speaking, their meaning is "if an individual is proved to be an instance

of C, then conclude that it is also an instance of D". Therefore they can be viewed as implications for which the contrapositive does not hold. A procedural rule $C \mapsto D$ can be formalized in $\mathcal{ALCK}$ by the epistemic sentence

$$\mathbf{K}C \sqsubseteq D$$

Notice that procedural rules can be regarded as a weak form of concept definitions.

A knowledge base in which the epistemic operator occurs only in rules of the above form has a unique preferred model; moreover, in such a case the entailment problem can be solved by constructing a knowledge base, called *first-order extension* [Donini *et al.*,1992; 1994]. Informally, the first-order extension is incrementally built by applying the following procedure: For each individual i explicitly mentioned in the ABox of the knowledge base and for each rule $C \mapsto D$, if $C(i)$ is a consequence of the knowledge base, then add $D(i)$ to the knowledge base. The first-order extension thus constructed can be used for answering queries in place of the epistemic knowledge base.

4 REASONING ABOUT ACTIONS: OUR PROPOSAL

In this section we present our framework for representing dynamic systems and reasoning about them. We first introduce the representation and, subsequently, address the reasoning method.

4.1 REPRESENTATION

Let us now describe how we use epistemic DLs to formalize dynamic systems. As we want to tackle the computational cost of reasoning, two aspects must be carefully considered:

1. The expressivity of the language, i.e. the set of constructs allowed;
2. The form of the axioms, i.e. the form of the inclusion assertions.

The research in DLs has shown that there is a trade-off between expressivity and complexity of reasoning, and has devised a number of languages for which reasoning without inclusion assertions is polynomial. However, adding general inclusions of the form $C \sqsubseteq D$ makes reasoning EXPTIME-hard even for a simple language as $\mathcal{FL}_0$ [McAllester,1991], which contains only intersection $\sqcap$ and universal quantification $\forall$. Hence, restrictions on the form of the inclusions are normally considered. In particular, cycles (recursive inclusions) [Nebel,1991; Buchheit *et al.*,1993; 1994; Calvanese,1996] are especially problematic from the computational point of view, and typically are not allowed.

Taking into account the above considerations, we model static axioms as acyclic inclusion assertions, and we model the dynamic axioms, for which the acyclicity condition would be too restrictive, by making a special use of procedural rules. In this way, the dynamic axioms *cannot be used* in the reverse direction for *contrapositive reasoning*, and this weakening allows for lowering the computational cost of reasoning in our formalism.

Specifically, in our ontology, an agent is, at any given point, in a certain *state*, represented by an *individual* in the domain of interpretation. *Properties of states* are represented as *concepts* of DLs. That is, a concept denotes a property that may hold in a state. *Actions*, which are assumed to be *deterministic*, are represented as *functional roles* – i.e. roles interpreted as functions instead of relations.

In fact, we distinguish two kinds of roles: *Static-roles*, which represent the usual notion of role in DLs and can be useful for structuring properties of states, and *action-roles*, which are functional roles that denote actions and are used in a special way.

The behavior of the agent is described by means of both static axioms and dynamic axioms. We formalize static axioms as acyclic inclusion assertions, not involving action-roles. Whereas, by exploiting the epistemic interpretation of procedural rules [Donini *et al.*,1994; 1995], we formalize dynamic axioms through epistemic sentences of the form:

$$\mathbf{K}C \sqsubseteq \exists \mathbf{K}R.\top \sqcap \forall R.D \qquad (2)$$

which can be intuitively interpreted as: For all possible interpretations, if a state (individual) x is an instance of C in all possible interpretations, then there exists a state y which is the (unique) R-successor of x in all possible interpretations, and y is an instance of D. In other words, for all possible transition graphs, if a state x satisfies the property C in all possible transition graphs, then there exists a state y which is the R-successor of x in all possible transition graphs, and y satisfies D. From now on we refer to states whenever the individuals are interpreted as states.

Notably, by using dynamic axioms of this special form, we recover the ability of representing the behavior of the system by means of a *single* graph – an ability that is generally lost when using other forms of dynamic axioms. Such a graph, which we may call *partial transition graph*, summarizes the common part of all transition graphs that, by our (partial) knowledge about the dynamic system, are considered possible. The partial transition graph gives us a description of a transition graph which is partial, in the sense that:

1. Certain states and transitions may be missing;
2. The properties of the states in the graph may be only partially specified.

Given an initial state satisfying certain properties, a plan exists for a specified goal if there exists a finite sequence of actions that, from the initial state, leads to a state satisfying the goal, regardless of which of the possible interpretations corresponds to the actual world. This condition is expressed by a logical implication similar to (1), namely:

$$\langle \Gamma_S \cup \Gamma_D, \{S(\mathit{init})\}\rangle \models ((\exists \mathbf{K}\alpha)^*.\mathbf{K}G)(\mathit{init}) \qquad (3)$$

where: (i) Γ_S and Γ_D respectively indicate the sets of static axioms and dynamic axioms; (ii) *init* names an individual representing the initial state, and S is a concept describing our knowledge about such an initial state; (iii) $(\exists \mathbf{K}\alpha)^*.\mathbf{K}G$ stands for *any* concept expression of the form

$$\exists \mathbf{K}R_1.\exists \mathbf{K}R_2.\ldots.\exists \mathbf{K}R_n.\mathbf{K}G$$

in which $n \geq 0$ and each R_i is an action-role, and it expresses the fact that from the initial state *init* there exists a sequence of successors (the same in every interpretation) that terminates in a state (the same in every interpretation) where G holds (in every interpretation). Intuitively, condition (3) checks for the existence of a state of the partial transition graph, reachable from the initial one, in which the goal is satisfied.

Observe that condition (3) holds iff for each preferred model $\mathcal{W}$ for $\Sigma = \langle \Gamma_S \cup \Gamma_D, \{S(\mathit{init})\}\rangle$, there exists a state $x \in \Delta$ such that $x \in G^{\mathcal{J},\mathcal{W}}$ for all $\mathcal{J} \in \mathcal{W}$. Indeed, by the special form of the dynamic axioms, such a state exists iff it is linked to the initial state by a chain of $\mathbf{K}R_i$, i.e. if there exists a sequence of successors (the same in every possible interpretation) that terminates in x.

4.2 REASONING

Let us now turn our attention to the problem of computing the entailment (3). We point out that in general the $\mathcal{ALCK}$-knowledge base Σ has many preferred models, which are distinguishable even up to renaming of individuals. Nevertheless, due to the special form of the epistemic sentences corresponding to the dynamic axioms in Σ, we can build the so-called *first-order extension* (FOE) [Donini *et al.*,1994] of the knowledge base $\Sigma = \langle \Gamma_S \cup \Gamma_D, \{S(\mathit{init})\}\rangle$, which consists of the knowledge base $\langle \Gamma_S, \{S(\mathit{init})\}\rangle$ augmented by the assertions which are consequences (up to renaming of individuals) of the epistemic sentences describing the dynamic axioms. The FOE of Σ provides a unique characterization of the knowledge that is shared by all the preferred models of Σ. In fact, the notion of FOE formalizes the concept of partial transition graph introduced above, which describes all common properties of the possible behaviors of the system.

The FOE of Σ, written $FOE(\Sigma)$, is computed by the algorithm shown in Fig. 4, in which

$$POST(\Sigma, R, s) = \{D_i \mid (\mathbf{K}C_i \sqsubseteq \exists \mathbf{K}R.\top \sqcap \forall R.D) \in \Sigma \wedge \Sigma \models C_i(s)\}$$

denotes the effect of the application of the rules of Σ involving the action-role R to the state s, namely the set of postconditions (concepts) of the rules which are triggered by s, and

$$CONCEPTS(\Sigma, i) = \{D \mid \Sigma \models D(i)\}$$

denotes the set of concepts verified by the state i in Σ.

Informally, the algorithm, starting from the initial state *init*, applies to each named state the rules in the set Γ_D which are triggered by such a state. A new state is thus generated, unless a state with the same properties had already been created. In this way the effect of the rules is computed, obtaining a sort of "completion" of the knowledge base.

It is easy to see that the FOE is unique, that is, every order of extraction of the states from the set STATES produces the same set of assertions, up to renaming of states. Moreover, it is easy to see that the algorithm terminates, that is, the condition STATES $= \emptyset$ is eventually reached, since the number of states generated is bounded to the number of different conjunctions of postconditions of the rules, i.e. 2^n, where n is the number of rules in Σ. Finally, the condition

$$CONCEPTS(\langle \Gamma_S, ABOX\rangle, l) = CONCEPTS(\langle \Gamma_S, ABOX'\rangle, j)$$

can be checked by verifying whether for each concept C, obtained as a conjunction of the postconditions of the rules in Γ_D, $\langle \Gamma_S, ABOX\rangle \models C(l)$ iff $\langle \Gamma_S, ABOX'\rangle \models C(j)$.

We point out that while the notion of minimization of the properties of states, realized through the propagation of rules (i.e. only the properties which are necessarily implied are propagated), is also correctly captured by the minimal knowledge semantics of $\mathcal{ALCK}$, the minimization obtained in the FOE by always generating a new successor state does not have a direct counterpart in the semantics: This is the reason why Σ has multiple preferred models, whereas the FOE is unique.

The following property establishes that, wrt the entailment problem (3), the first-order extension of Σ represents the information which must hold in *any* preferred model for Σ.

Theorem 4.1 *There exists a state x such that*

$$FOE(\Sigma) \models G(x) \qquad (4)$$

if and only if, for each preferred model $\mathcal{W}$ for Σ, there exists a state x such that $x \in G^{\mathcal{J},\mathcal{W}}$ for all $\mathcal{J} \in \mathcal{W}$.

Sketch of the proof. If-part. Suppose that for each model $\mathcal{W}$ for Σ there exists a state x such that $x \in G^{\mathcal{J},\mathcal{W}}$ for all $\mathcal{J} \in \mathcal{W}$. Now, it is easy to see that there exists a preferred model $\mathcal{W}'$ for Σ such that for each state s in $FOE(\Sigma)$ there exists a state s'

```
ALGORITHM FOE
INPUT: Σ = ⟨Γ_S ∪ Γ_D, {S(init)}⟩
OUTPUT: FOE(Σ)
begin
  STATES = {init};
  ALL-STATES = {init};
  ABOX = {S(init)};
  repeat
    s = choose(STATES);
    for each action-role R do
    begin
      s' = NEW state name;
      ABOX' = ABOX ∪ {R(s, s')} ∪ {D_i(s') | D_i ∈ POST(⟨Γ_S ∪ Γ_D, ABOX⟩, R, s)};
      if there exists a state s'' ∈ ALL-STATES such that
        CONCEPTS(⟨Γ_S, ABOX⟩, s'') = CONCEPTS(⟨Γ_S, ABOX'⟩, s')
      then ABOX = ABOX ∪ R(s, s'')
      else begin
             ABOX = ABOX';
             STATES = STATES ∪{s''}
             ALL-STATES = ALL-STATES ∪{s''}
           end
    end;
    STATES = STATES −{s}
  until STATES = ∅;
  return ⟨Γ_S, ABOX⟩
end;
```

Figure 4: Algorithm computing $FOE(\Sigma)$

in $\mathcal{W}$ isomorphic to s, that is, for every concept C, $FOE(\Sigma) \models C(s)$ iff $s' \in C^{\mathcal{J},\mathcal{W}}$. Therefore, the existence of such a model $\mathcal{W}'$ implies the existence of a state y such that $FOE(\Sigma) \models G(y)$.

Only-if-part. Assume there exists a state x such that $FOE(\Sigma) \models G(x)$. Then, there is a finite sequence of actions $R_{i_1}, \ldots, R_{i_n}$ (the plan) that generates x. Let x_1 be the R_{i_1}-successor of *init* in $FOE(\Sigma)$, and let $\mathcal{W}$ be any preferred model for Σ. Now, the properties that *init* is known to verify in all worlds of $\mathcal{W}$ are at least the properties stated in the initial situation. Consequently, the set of epistemic sentences, corresponding to the dynamic axioms whose antecedent is satisfied by *init* in $FOE(\Sigma)$, implies the same set of properties on another state (say y_1) that is the same in each interpretation $\mathcal{J}$ of $\mathcal{W}$. That is, y_1 satisfies *at least* the same properties satisfied by x_1. Now, let x_2 be the R_{i_2}-successor of x_1. The same kind of reasoning can be applied, thus showing that there must exist a state y_2 in $\mathcal{W}$ satisfying in each world at least the same properties verified by x_2. By iteration we conclude that there exists a state y_n such that $x \in G^{\mathcal{J},\mathcal{W}}$ for all $\mathcal{J} \in \mathcal{W}$, which concludes the proof. □

By the above property, we can solve the planning problem (3) by verifying whether there is an x in $FOE(\Sigma)$ that satisfies the goal G.

4.3 COMPLEXITY

As for the computational aspects of reasoning about actions in the epistemic framework based on $\mathcal{ALCK}$, it turns out that the planning problem is PSPACE-complete which is a direct consequence of Theorem 4.1 and of the following property.

Theorem 4.2 *The problem of establishing whether there exists a state x such that $FOE(\Sigma) \models G(x)$ is PSPACE-complete.*

Sketch of the proof. PSPACE-hardness follows from the fact that the subsumption problem for acyclic $\mathcal{ALC}$ TBoxes, which is PSPACE-complete [Calvanese,1996], can be reduced ($\Gamma_D = \emptyset$) to the problem of establishing whether there exists a state x such that $FOE(\Sigma) \models G(x)$. Membership in PSPACE is due to the fact that the maximum number of states generated in $FOE(\Sigma)$ is 2^n (where n is the number of dynamic axioms), therefore, if there exists a state x in $FOE(\Sigma)$ such that $G(x)$ holds, then such a state can be generated through a sequence of actions whose length is less or equal to 2^n. This property allows for the generation of all the states, one at a time, using a polynomial amount of space. □

Notice that the algorithm for computing $FOE(\Sigma)$ uses exponential space, because $FOE(\Sigma)$ can be used to

find a plan, not only to know whether there exists a plan. It is easy to modify the above algorithm in order to answer to the plan existence problem using polynomial space only.

Let us now compare our formalization with the one we get leaving out the epistemic operator (i.e. Rosenschein's formalization), hence using $\mathcal{ALC}$ instead of $\mathcal{ALCK}$. The difference between the two formalizations lies in the different representation of the dynamic axioms Γ_D, which are formalized through epistemic sentences in $\mathcal{ALCK}$ of the form (2), whereas they are expressed by ordinary axioms (implications) in $\mathcal{ALC}$, of the following form:

$$C \sqsubseteq \exists R.\top \sqcap \forall R.D$$

corresponding to Rosenschein's dynamic axioms

$$C \Rightarrow \langle R \rangle \mathtt{tt} \wedge [R]D$$

The planning problem (1) is expressed in such a setting by the following entailment problem:

$$\langle \Gamma_S \cup \Gamma_D, \{S(\mathit{init})\} \rangle \models ((\exists \alpha)^*.G)(\mathit{init}) \qquad (5)$$

We now show that the use of procedural rules in the formalization of the dynamic axioms actually weakens the deductive capabilities of the agent formalized. This can be explained by the following simple example. Suppose we have the following dynamic axioms:

$$C \sqsubseteq \exists R.\top \sqcap \forall R.D$$

$$\neg C \sqsubseteq \exists R.\top \sqcap \forall R.D$$

and suppose the goal is D and the initial situation does not specify the truth value of C. It is easy to see that in the $\mathcal{ALC}$ formalization (corresponding to Rosenschein's framework) the answer to the planning problem is yes (the desired plan consists of the action R), while in the $\mathcal{ALCK}$ framework the answer to the planning problem is no.

This is precisely due to the different formalization of the dynamic axioms: In the $\mathcal{ALC}$ setting, the agent is able to conclude that he is able to perform action R, since in the current *state of the world* either C or $\neg C$ holds. Conversely, in the $\mathcal{ALCK}$ framework, in the *epistemic state* of the agent neither C nor $\neg C$ holds, since the agent *does not know* the truth value of C, therefore he concludes that he is not able to perform action R.

Let us now slightly modify the above example. Suppose we have the following dynamic axioms:

$$C \sqsubseteq \exists R_1.\top \sqcap \forall R_1.D$$

$$\neg C \sqsubseteq \exists R_2.\top \sqcap \forall R_2.D$$

and again suppose that the goal is D and the initial situation does not specify the truth value of C. Now, there is no *known* sequence of actions leading to the state satisfying the goal, yet in the $\mathcal{ALC}$ setting the answer to the planning problem is yes, whereas it is still no in the epistemic framework.

This example highlights the fact that in the $\mathcal{ALCK}$ setting *only constructive proofs are taken into consideration*, in the sense that the entailment (3) holds only if there exists a *known* sequence of actions leading to the state satisfying the goal. That is, if (3) holds then we are always able to extract an effective plan.

Conversely, it is easy to see that, if in the epistemic formalization of actions the entailment (3) holds, then the entailment (5) holds in the $\mathcal{ALC}$ setting. Therefore, with respect to the planning problem, the epistemic framework based on $\mathcal{ALCK}$ is a *sound and incomplete* approximation of the non-epistemic one. That is, the use of the epistemic operator in the formalization of actions allows for a *principled* weakening of the deductive capabilities of the agent.

5 THE MOBILE ROBOT "TINO"

Our approach to reasoning about actions has been implemented on the *Erratic* base [Konolige,1995]. The robot, that, as mentioned, has been named Tino, has been successfully tested on several real and simulated office environments.

Tino is based on a two-level architecture, which combines a reactive control mechanism with a planning system. The idea dates back to the architecture of the robot Shakey, in which the planning system was STRIPS. However, the Erratic base allows for an effective combination of both horizontal and vertical decomposition (see [Brooks,1986]). By a horizontal decomposition the system can react immediately in dynamic environments. While a vertical decomposition is in the relationship between the module which is responsible for planning and the underlying fuzzy controller. Each of these modules has its own representation of the environment and of the plan. The communication between the two modules is realized by a plan execution module. Thus, as in [Gat,1992], we have a heterogeneous architecture, since we have implemented our planning system and then we have connected it with the existing controller.

Below we sketch the basic elements of the implementation of the planning component and of the plan execution component.

5.1 PLAN GENERATION

One of the motivations underlying our proposal for reasoning about actions is the possibility of relying on a knowledge representation system based on DLs for the implementation. In particular, we have chosen CLASSIC [Borgida *et al.*,1989], a well-known system based on DLs, to take advantage of an efficient and reliable reasoning system. However, the language for

representing knowledge is less expressive than the DL we have considered so far. Nonetheless, we obtain an interesting setting where the plan can be generated in polynomial time.

More specifically, we use a subset of the language $\mathcal{ALCK}$, corresponding to some of the constructs available in CLASSIC, that we write for ease of notation using $\sqcap$ for **AND**, $\forall$ for **ALL** and $\exists R.\{a\}$ for **FILLS**.

Static axioms are expressed either as inclusion assertions or as concept definitions, written $\doteq$ and interpreted as necessary and sufficient conditions (see for example [Buchheit *et al.*,1994]). In both cases cycles are not allowed. Dynamic axioms are represented as CLASSIC *rules* denoted with $\mapsto$.

Each rule is thus written as

$$C \mapsto \exists R.\{a\} \sqcap \forall R.D$$

and can be read as follows: For each named individual i classified under C, connect i to the individual a through the role R, and classify a under D. Therefore a is an individual denoting the state reached as a result of the execution of action R.

Notice that we are forced to use the **FILLS** (written $\exists R.\{a\}$) construct because we cannot express $\exists \mathbf{K}R.\top$ in CLASSIC. This gives us a sound implementation as long as for each action R the preconditions C in dynamic axioms involving R are disjoint from each other. Indeed, this condition implies that in every state at most one of the preconditions C for each action R is satisfied, consequently the only concept that holds in an R-successor, obtained by applying the dynamic axiom, is D.

With the above described representation, computing the FOE is done in polynomial time, because the number of individuals is at most linear in the number of rules, and the condition for the application of a rule can be checked in polynomial time. Notice that we cannot compute the FOE of a knowledge base Σ in the general case using CLASSIC, because there is no other way to state the existence of a named individual (representing a successor state) other than explicitly naming it through the construct **FILLS**.

The plan is extracted by the explanation facility provided with the rule mechanism of the system, which allows for an automatic generation of a graph (essentially a part of the FOE) with all the paths from the initial state to the states that satisfy the goal. The plan to be sent to the robot is then selected by finding the path (between the initial state and the states that satisfy the goal) which is minimal in terms of the number of transitions (actions).

Example 5.1 Given the map shown in Fig. 5, referring to an office environment constituted by rooms, doors and corridors, we can describe it through the following knowledge base:

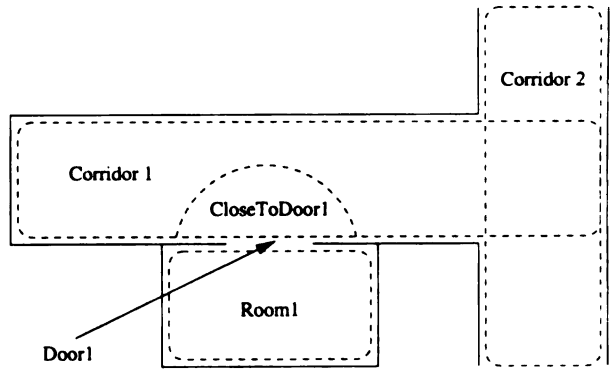


Figure 5: A simple environment

$$\begin{array}{lcl}
Corridor1 & \sqsubseteq & Corridor \\
Corridor2 & \sqsubseteq & Corridor \\
Room1 & \sqsubseteq & Room \\
CloseToDoor & \doteq & Corridor \sqcap \forall NextDoor.Door \sqcap \\
 & & \exists NextDoor.\top \\
CloseToDoor1 & \doteq & CloseToDoor \sqcap Corridor1 \sqcap \\
 & & \exists NextDoor.\{xDoor1\}
\end{array}$$

$$\begin{array}{lcl}
Corridor1 & \mapsto & \exists FollowC1ToD1.\{xCloseToDoor1\} \sqcap \\
 & & \forall FollowC1ToD1.CloseToDoor1 \\
Corridor1 & \mapsto & \exists FollowC1ToC2.\{xCorridor2\} \sqcap \\
 & & \forall FollowC1ToC2.Corridor2 \\
CloseToDoor1 & \mapsto & \exists EnterD1.\{xRoom1\} \sqcap \\
 & & \forall EnterD1.Room1 \\
Room1 & \mapsto & \exists ExitD1.\{xCloseToDoor1\} \sqcap \\
 & & \forall ExitD1.CloseToDoor1
\end{array}$$

$$Door(xDoor1)$$
$$Corridor1(xCorridor1)$$

Let us now highlight the use of static axioms for the description of knowledge about environments and for the classification of states. In the above example, the concept *CloseToDoor* formalizes the property of being in a corridor and close to a door, while the concept *CloseToDoor1* represents the property of being close to a particular door (*Door1*). *NextDoor*, which is assumed to be a functional role, is used to describe a static property of a portion of a corridor.

The action graph relative to the above knowledge base is given in Fig. 6. Notice that with the rule propagation mechanism we can produce many edges of the graph with a single rule. In fact, suppose there are many doors in *Corridor1*, with the only rule that describes the action $FollowC1ToC2$, we can connect in the graph all the states $xCloseToDoor(i)$ to the state $xCorridor2$ with edges labeled with the action $FollowC1ToC2$. This is due to the fact that we can write static axioms such that $CloseToDoor1 \sqsubseteq Corridor1$.

Moreover, static axioms allow us to write postconditions of actions in a simple way. For example in the de-

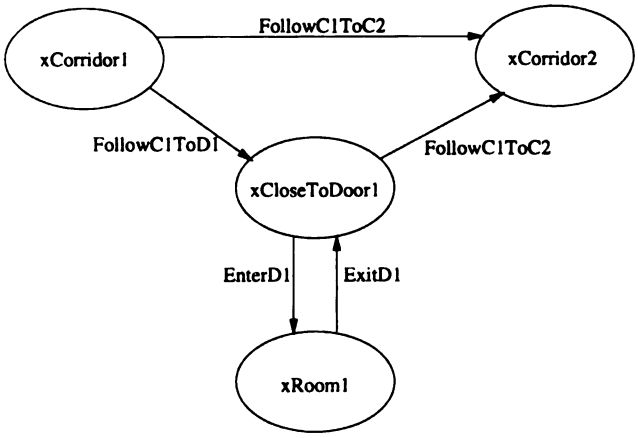


Figure 6: The action graph

scription of the action $ExitD1$ we have only mentioned $CloseToDoor1$ as postcondition, instead of writing all the postconditions (in this case also $Corridor1$) explicitly.

The ability to provide a taxonomic representation of the environment not only provides a more compact way to specify the knowledge about the problem domain, but also to have a more flexible and easy to modify representation. In particular, we have taken advantage of these features both in modeling different environments and in the implementation of a module that takes as input a topological representation of the map and generates a CLASSIC knowledge base.

5.2 PLAN EXECUTION

The planning system is activated by a plan execution module which provides the connection to the robot software. The control of the robot is achieved by means of a fuzzy controller [Saffiotti *et al.*,1995] which takes care of obstacle avoidance while the robot is trying to achieve a high-level goal such as reaching the next door in the corridor.

More specifically, the plan execution module turns the sequence of actions, expressed as CLASSIC roles, into a specification for the control software. The control system drives the robot low-level movement commands by means of control schemata, called *behaviors*, which specify both how to implement high-level actions and how to perform several kinds of reactive control, such as obstacle avoidance. Therefore, the simultaneous activation of reactive behaviors, and of the ones implementing actions is usually requested to perform a task. For example, the activation of the behaviors *Avoid Obstacle*, *Keep Off* and *Follow Corr*, is used to realize navigation within a corridor. The actual execution can be viewed as the *blending* of such behaviors, based on the information acquired through the sensors (see [Saffiotti *et al.*,1995] for further details). The idea is related to the notion of plan-as-communication [Agre and Chapman,1990], since the plan is not mechanically executed, but is used to decide which behavior to activate.

As each module has its own representation of information, the exchange of information between the two layers is a critical aspect. The planning system sends a plan to the control system, the latter can detect a plan failure and reply to the former with a justification for the failure (such as "door closed") so that the planning system can re-plan after updating the knowledge base.

Example 5.2 We can describe the fact that the robot can enter in a room only if it is close to an open door by adding the following declarations to the knowledge base in Example 5.1.

$$\begin{aligned} CloseToOpenDoor &\doteq CloseToDoor \sqcap \\ &\quad \forall NextDoor.Open \\ CloseToOpenDoor1 &\doteq CloseToOpenDoor \sqcap \\ &\quad CloseToDoor1 \end{aligned}$$

$$\begin{aligned} CloseToOpenDoor1 \mapsto\ & \exists EnterD1.\{xRoom1\} \sqcap \\ & \forall EnterD1.Room1 \end{aligned}$$

$$Door(xDoor1)$$
$$Open(xDoor1)$$

Now, in presence of a failure during a plan execution, caused by the fact that door $xDoor1$ is closed, it is sufficient to update the knowledge base, by retracting the assertion $Open(xDoor1)$, so that the new rule involving *EnterD1* cannot be applied and the associated edge will not be in the action graph. If there is another door to access the room, it may be possible to select an alternative plan to reach it.

6 CONCLUSIONS

The goal of our work was to devise a principled and practically feasible realization of a KR&R approach to reasoning about actions in the realm of mobile robots. In the paper we have presented a formal setting for reasoning about actions and its implementation on the *Erratic* base, integrating reacting behavior with action planning and execution.

We believe that the results of our work are twofold. From the standpoint of the research on reasoning about actions, the basis for our work has been the formal correspondence between Propositional Dynamic Logics and Description Logics. This has lead us not only to a semantically justified implementation, but also to the adaptation of the formal setting in new and interesting directions. In particular, we have shown that the use of rules instead of axioms both reduces the computational complexity of the planning problem and (under some restrictions) allows for the implemen-

tation of the theoretical framework for reasoning about actions in an efficient KR system (CLASSIC).

As compared with other approaches to action planning for mobile robots, the choice of a Knowledge-Representation System, together with the associated methodology for representing the dynamic environment and reasoning about actions, has led to a very flexible implementation of the planning component, that can be easily adapted to new environments and accommodate the changes in the environment. To this end, the possibility of structuring the representation of environments using the features of a DL representation language plays a crucial role.

The implementation developed so far is mainly concerned with the position of the robot and its movement capabilities. We are currently working at several extensions of the proposed framework, that will enable Tino to address more complex scenarios. To this end, we can exploit the notion of epistemic state of the agent, using it to address several issues arising in complex dynamic domains. In particular, we are currently focusing on the following aspects:

1. **Frame problem.** It turns out [Donini *et al.*,1995] that a slight modification in the semantics of the epistemic operator allows for the representation of default rules in $\mathcal{ALCK}$, thus allowing for the formalization of the notion of default persistence of knowledge, realized through the use of defaults. This property, together with the notion of default persistence of ignorance encoded in the semantics, allows for a formalization of the commonsense law of inertia, realized through a "small" number of epistemic axioms (formalizing default rules).
2. **Sensing.** The possibility of representing the epistemic state of the agent allows for a very simple formalization of sensing (or knowledge-producing) actions. Indeed, the semantic of the epistemic operator in $\mathcal{ALCK}$ embodies the principle of minimal learning, or default persistence of ignorance [Scherl and Levesque,1993], thus allowing for a simple treatment of actions whose execution only changes the knowledge of the agent without affecting the state of the world.

Acknowledgments

The work has been supported by Italian MURST. Riccardo Rosati acknowledges the AI Center of SRI International, Menlo Park, for the support provided during his visiting period.

References

[Agre and Chapman, 1990] Philip E. Agre and David Chapman. What are plans for? *Robotics and Autonomous Systems*, 6:17–34, 1990.

[Artale and Franconi, 1994] Alessandro Artale and Enrico Franconi. A computational account for a description logic of time and action. *Proceedings of the Fourth International Conference on the Principles of Knowledge Representation and Reasoning (KR-94)*, 1994.

[Borgida *et al.*, 1989] Alexander Borgida, Ronald J. Brachman, Deborah L. McGuinness, and Lori Alperin Resnick. CLASSIC: A structural data model for objects. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 59–67, 1989.

[Borgida, 1992] Alex Borgida. Towards the systematic development of description logic reasoners: Clasp reconstructed. *Proceedings of the Third International Conference on the Principles of Knowledge Representation and Reasoning (KR-92)*, 1992.

[Brooks, 1986] Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1), 1986.

[Buchheit *et al.*, 1993] Martin Buchheit, Francesco M. Donini, and Andrea Schaerf. Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research*, 1:109–138, 1993.

[Buchheit *et al.*, 1994] Martin Buchheit, Francesco M. Donini, Werner Nutt, and Andrea Schaerf. Terminological systems revisited: Terminology = schema + views. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 199–204, Seattle, USA, 1994.

[Calvanese, 1996] Diego Calvanese. Reasoning with inclusion axioms in description logics: Algorithms and complexity. In W. Wahlster, editor, *Proceedings of the Twelfth European Conference on Artificial Intelligence (ECAI-96)*, pages 303–307. John Wiley & Sons, 1996.

[De Giacomo and Lenzerini, 1994] Giuseppe De Giacomo and Maurizio Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proceedings of the 12th National Conference on Artificial Intelligence*, pages 205–212, 1994.

[De Giacomo and Lenzerini, 1995a] Giuseppe De Giacomo and Maurizio Lenzerini. Enhanced propositional dynamic logic for reasoning about concurrent actions (extended abstract). In *Working notes of the AAAI 1995 Spring Symposium on Extending Theories of Action: Formal and Practical Applications*, pages 62–67, 1995.

[De Giacomo and Lenzerini, 1995b] Giuseppe De Giacomo and Maurizio Lenzerini. PDL-based framework for reasoning about actions. In *Proceedings*

*of the Fourth Conference of the Italian Association for Artificial Intelligence (AI*IA-95)*, number 992 in Lecture Notes In Artificial Intelligence, pages 103–114. Springer-Verlag, 1995.

[Donini *et al.*, 1992] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Werner Nutt, and Andrea Schaerf. Adding epistemic operators to concept languages. In *Proceedings of the Third International Conference on the Principles of Knowledge Representation and Reasoning (KR-92)*, pages 342–353. Morgan Kaufmann, Los Altos, 1992.

[Donini *et al.*, 1994] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Werner Nutt, and Andrea Schaerf. Queries, rules and definitions. In *Foundations of Knowledge Representation and Reasoning*. Springer-Verlag, 1994.

[Donini *et al.*, 1995] Francesco M. Donini, Daniele Nardi, and Riccardo Rosati. Non first-order features in concept languages. In *Proceedings of the Fourth Conference of the Italian Association for Artificial Intelligence (AI*IA-95)*, Lecture Notes In Artificial Intelligence. Springer-Verlag, 1995.

[Gat, 1992] Erann Gat. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. *Proceedings of the 11th National Conference on Artificial Intelligence*, 1992.

[Kaelbling and Rosenschein, 1995] L. Kaelbling and S. Rosenschein. A situated view of representation and control. *Artificial Intelligence*, 73:149–173, 1995.

[Koehler, 1994] Jana Koehler. An application of terminological logics to case-based reasoning. *Proceedings of the Fourth International Conference on the Principles of Knowledge Representation and Reasoning (KR-94)*, 1994.

[Konolige, 1995] Kurt Konolige. Erratic competes with the big boys. *AAAI Magazine*, Summer:61–67, 1995.

[Kozen and Tiuryn, 1990] D. Kozen and J. Tiuryn. Logics of programs. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 790–840. Elsevier Science Publishers (North-Holland), Amsterdam, 1990.

[Lesperance *et al.*, 1994] Y. Lesperance, H.J. Levesque, F. Lin, D. Marcu, R. Reiter, and R.B. Scherl. A logical approach to high-level robot programing. In *AAAI FAll Symposium on Control of the Physical World by Intelligent Systems*, 1994.

[McAllester, 1991] David McAllester, 1991. Unpublished Manuscript.

[Nebel, 1991] Bernhard Nebel. Terminological cycles: Semantics and computational properties. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 331–361. Morgan Kaufmann, Los Altos, 1991.

[Reiter, 1993] R. Reiter. Proving properties of states in the situation calculus. *Artificial Intelligence*, 64:337–351, 1993.

[Rosenschein, 1981] S. Rosenschein. Plan synthesis: a logical approach. In *Proc. of the 8th Int. Joint Conf. on Artificial Intelligence*, 1981.

[Saffiotti *et al.*, 1995] A. Saffiotti, K. Konolige, and E. Ruspini. A multivalued logic approach to integrating planning and control. *Artificial Intelligence*, 76:481–526, 1995.

[Scherl and Levesque, 1993] Richard Scherl and Hector J. Levesque. The frame problem and knowledge producing actions. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pages 689–695, 1993.

[Schild, 1991] Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 466–471, Sydney, 1991.

[Simmons, 1992] Reid Simmons. An architecture for coordinating planning, sensing and action. *Proceedings of the 11th National Conference on Artificial Intelligence*, 1992.

Spatial Representation and Reasoning

Semantical Foundations of Spatial Logics

Oliver Lemon*
Dept. of Computer Science
University of Manchester,
Manchester, M13 9PL
lemonoj@cs.man.ac.uk

Abstract

We explore several "spatial" logics and investigate their credentials as logics *of* space or of spatial objects and relations. A semantical adequacy criterion for spatial logics is developed, according to which a logic is *spatial* only if consistent theories in that logic are realizable in a standard model of space. Various (so-called) spatial logics are shown not to satisfy this criterion. In effect, these observations amount to incompleteness results for classes of spatial logics, because they show that consistent sets of formulae in these logics have no models of the intended sort. In addition, we present a complete axiomatization of space; a modal logic of connected regions of the plane.

1 MOTIVATION

In recent years there has been much interest in logics of space (eg: [Randell *et al.*, 1992, Gotts *et al.*, 1996, Asher and Vieu, 1995, Bennett, 1995]).
Different research programmes have developed from work (eg: [Tarski, 1956, Whitehead, 1929, Clarke, 1981, Clarke, 1985]) on logic and mereotopology [Biacino and Gerla, 1991], of [Allen, 1981] on temporal logic, and investigations into modal logics of space [Rescher and Garson, 1968, von Wright, 1979], resulting in a lively research area. Logics of space have obvious applications in Knowledge Representation and Reasoning [Randell *et al.*, 1992, Bennett, 1994, Pratt, 1993, Hernández, 1994, Nebel, 1995, Bennett, 1995], Robotics, Spatial Information Theory [Frank and Campari, 1993, Frank and Kuhn, 1995], and the theory of Geographical Information Systems (GIS) [Worboys, 1991, Mark and Frank, 1991, Jeansoulin and Mathieu, 1995].

An important part of the motivation for developing spatial logics lies in a dissatisfaction with the ontological richness of the standard Cartesian model of space as a collection of tuples of real numbers. The possibility of infinite collections of regions all occupying the same volume, and the existence of pathological regions with counterintuitive properties (as exemplified by the infamous Banach-Tarski theorem [Wagon, 1993]) suggest that this model provides more than our intuitive concepts of space require. Perhaps, then, there is a simpler rational reconstruction of these notions—a model of space more suited to the sorts of distinctions that we feel happy about making. Perhaps such a simpler model also has the more practical advantage of suggesting efficient procedures for spatial inference, forwarding another important computational motivation.

One way to construct a simpler model of space, of course, is to devise a logic containing operators with a spatial gloss. Just as, in temporal logic, one has operators such as "at some time in the past", "always in the future," "since", "until" , so in spatial logic one might have operators (or, alternatively, relations or predicates) with interpretations such as "somewhere", "at all points in the vicinity", "at some neighbouring point" and so on. By writing down axioms governing these operators, and using standard results concerning the existence of models for consistent theories in modal and first-order logics, one achieves alternative models of space to the Cartesian model.

The general problem with these "spatial" logics, however, is that it is sometimes hard to know whether one has written down enough axioms to characterize the spatial notions in question. Another way of putting the same worry is that, while a collection of suggested

*See http://www.cs.man.ac.uk/ai/oliver/mapsem.html for related papers and discussion.

axioms may well have models, it is sometimes hard to see why we should regard these models as models of space at all. This paper sets out a criterion which proposed spatial logics should fulfil to be counted as spatial; we also show that various proposed spatial logics in the literature fail to do so. In addition, we present a logic which meets our proposed criterion; a logic of connected regions of the plane, and we highlight the importance of *extended* modal logics [de Rijke, 1992] for the project of giving intensional models of space.

2 MODAL SPATIAL LOGICS

Most spatial logics deal with spatial relationships in one of two ways: either they employ a modal operator to index propositions to locations, or they quantify explicitly over those locations. In the modal approach, an indexed 'position operator' P_α is added to classical propositional logic, so that $P_\alpha(p)$ can be interpreted variously as "proposition p is realized at position α" or "proposition p is realized at a position α units from here". Different logics may be constructed depending on the particular properties of P and the range of the parameter α, which can be quantified over. Positional, topological, and "spatial" intepretations of modal logics have been suggested by [Rescher and Garson, 1968, von Wright, 1979, Bennett, 1995, Worboys, 1991][1], and intensional models of topologies and geometries (in too abstract a sense for our purposes) have been constructed by [Moss and Parikh, 1992, Georgatos, 1994, Balbiani *et al.*, 1995]. Most familiar modal logics are finitely axiomatizable, decidable, and have the finite model property. They thus clearly present a promising framework in which to reason qualitatively about space. Rescher and Garson [Rescher and Garson, 1968] present a quite general "positional logic" in which temporal and spatial logics can be formulated.

A good example of a spatial logic – and of their potential shortcomings – is provided by von Wright [von Wright, 1979, von Wright, 1984]. Von Wright, employs the spatial interpretation of modal operators in order to analyse the logics of the spatial terms "near", "elsewhere", and "somewhere". In the same framework as [Rescher and Garson, 1968], von Wright considers various modal axioms in the 3 different cases. For example, the axiom $p \rightarrow \Diamond p$ holds only when the modal operator is interpreted as "somewhere". In this fashion, von Wright conjectures that the "near" relation (which is symmetric[2]) corresponds to the modal system SwB (weak Brouwer)[3], "elsewhere" corresponds to $SwB + (\Diamond \Diamond p \rightarrow p \vee \Diamond p)$, and that "somewhere" correponds to $S5$[4].

However, von Wright's logic of "near" (the weak Brouwer system) is inadequate. Consider the following schematic formula:

$$\bigwedge_{1 \leq i \leq n} \Diamond \Diamond \left(\Box \left(c_i \& \neg \left(\bigvee_{1 \leq j \leq n}^{j \neq i} c_j \right) \right) \right) \qquad (1)$$

Since the logic SwB is characterized by serial and symmetric Kripke frames, the diagram of figure 1 is a permissible SwB-structure making the formula (1) true in the case where $n = 6$. Clearly, there is nothing special about the number 6 here; any number of spokes could be added. Hence all instances of schema (1) are SwB-consistent.

But now consider the proposed spatial gloss on $\Diamond$, in which $\Diamond$ is interpreted as meaning "nearby". Given that all instances of schema (1) are SwB-consistent, it makes sense to ask whether the spatial arrangement they describe is really consistent given the intended interpretation of the operator. We can tackle this question formally by defining a "classical" or "intended" interpretation – let us call it M_2 – as follows. Let us say that the possible worlds of M_2 are the set of points in R^2 and the accessibility relation r is defined by

$$r(\langle x_1, y_1 \rangle, \langle x_2, y_2 \rangle)$$

$$\text{iff } (x_1 - x_2)^2 + (y_1 - y_2)^2 \leq d^2$$

where d is some fixed real $\neq 0$. Once we have a classical model, we can sensibly ask whether an instance of schema (1) is true anywhere in it. But in fact, for $n \geq 6$ there are no possible configurations of points in the plane which can serve as an interpretation of formula (1). For one would have to fit 6 or more non-overlapping closed circles of radius d around (overlapping the edge of) a closed circle of radius d—and it is routine to show that this situation is geometrically impossible. The situation is illustrated in figure 2. Thus, von Wright's proposed logic of near allows us to write down consistent formulae (consistent, that is, in his

[1] Indeed, they have also been discussed by logicians such as Robert Stalnaker, Nuel Belnap, and Krister Segerberg (see [von Wright, 1979, Rescher and Garson, 1968, von Wright, 1984]).

[2] Although intuition might dictate that it is also irreflexive, such constraints cannot be captured in normal modal systems, as we shall see.

[3] Its axioms are $\Diamond(p \vee q) \leftrightarrow \Diamond p \vee \Diamond q$ and $\Diamond \neg \Diamond p$. Weak modal logics lack the axiom $p \rightarrow \Diamond p$.

[4] These logics contain the "Theorem of Reversibility", $p \& \Diamond q \rightarrow \Diamond(q \& \Diamond p)$, which distinguishes them from temporal logics (in which it is not provable).

logic) which describe no realizable situation given the intended interpretation of the operator[5].

Of course, one might object to the formal reconstruction of the intended interpretation. Perhaps, for example, the intended interpretation—call it M_3—has, as its "possible worlds" the points of R^3, with the accessibility relation defined by:

$$r(\langle x_1, y_1, z_1\rangle, \langle x_2, y_2, z_2\rangle)$$

$$\text{iff } (x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2 \leq d^2$$

Here too, however, a similar problem arises; except that this time we must make $n \geq 12$ to achieve a consistent instance of schema (1) which is unrealizable in M_3.

Of course, one still might object to our proposed reconstruction. (Maybe the domain von Wright had in mind was Hilbert space.) But we need to be told what the correct interpretation of the logic is.

Technically, we have proved:

Theorem 1 *The logic SwB is not complete for the interpretations M_2 and M_3.*

Informally, this amounts to the following: Let the modal operator $\diamond$ in the logic SwB have the intended geometrical interpretation "at some point no more than distance d away". Then there are consistent sentences of the logic which have no models with the intended geometrical interpretation, in either 2 or 3 dimensions.

What do we learn from theorem 1? Just because we have an axiomatization which seems intuitively correct for some spatial notion, that does not mean that such a logic actually captures that notion. In order to do that, consistent sets of sentences in that logic must be realizable in R^2 or R^3 using the intended interpretations of the spatial notions the logic is supposed to axiomatize. In putting forward the logic SwB, von Wright has not provided us with a logic of "near" at all; he has just written down three or four of its properties. The situation is even worse in general; in order to capture the irreflexivity and intransitivity of many spatial relations, we need to use more expressive modal languages.

In general, these sorts of problems in the semantics of spatial logics motivate the following criterion.

[5]Note: if $\diamond$ were interpreted as "at some location *strictly* less than distance d from here", the same problem would arise.

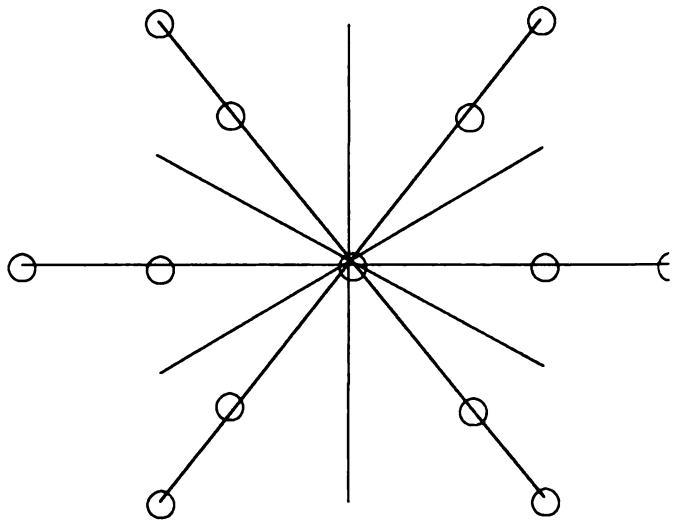


Figure 1: A Structure Realizing the Instance of the Schema (1) with $n = 6$

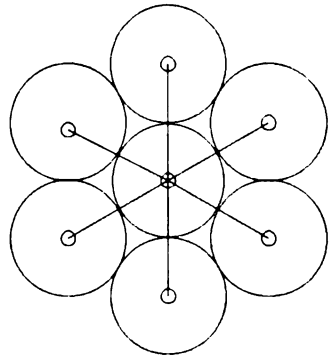


Figure 2: An Arrangement Depicting the Unrealizability of the Schema (1) when $n \geq 6$

3 AN ADEQUACY CRITERION

Theories expressed in spatial logics ought to be embeddable in classical models of space, otherwise they are not logics of space at all. In other words, a minimal requirement on a spatial description language is that it at least agree with the standard model of space.

Criterion 1 *A logic is spatial only if it is equipped with a "classical" or "intended" spatial interpretation or set of interpretations, with respect to which it is complete.*

What counts as a spatial interpretation is of course a vague matter. But that should not trouble us; what counts as a spatial logic is a vague matter too.

We shall demonstrate that such requirements can be met for a variety of simple modal spatial logics.

4 SOME ADEQUATE MODAL SPATIAL LOGICS

We have illustrated the problem of the inadequacy of a "spatial" logic, which fails adequately to characterize its intended interpretation. We now present some spatial logics which are adequate.

We begin with some trivial examples. "Elsewhere" and "somewhere" can be axiomatized using the modal logic DL_m (of "inequality" eg: [de Rijke, 1992]) and S5 respectively – the frame conditions for these logics are seriality, irreflexivity, and symmetry, and reflexivity, symmetry and transitivity respectively. It is a trival matter to see that such logics are adequate. For modal theories where $\Diamond p$ is interpreted as "elsewhere p", for example, simply take each new location to be some point in R^2 which is distinct from all the previous points, and interpret each "elsewhere p" formula simply as existence of such a new point. For the case of "somewhere" one simply drops the requirement that newly chosen points are distinct.

The above results give rise to some interesting observations on the expressive power of "spatial" languages. For example, it seems that talk of *spatial* modal logics is too strong – they are only expressive enough to capture *positional* constraints imposed by spatial structure. This is not so surprising when one considers the Kripke semantics for modal logics – given in terms of indexed points. Such a weak structure could hardly be expected to capture the richness of the geometric plane. In this regard, the comments of [Rescher, 1979], in reply to [von Wright, 1979], seem prophetic. Indeed, it is well known (eg: [van Benthem, 1984]) that modal logics cannot capture such structural phenomena as irreflexivity and intransitivity. In addition, a complete description of many spatial structures requires reference to *unique* positions, thus demanding the introduction of *nominals* for locations in the appropriate languages. An interesting application of the extended modal systems of [de Rijke, 1992] lies in their ability to express both nominals and such constraints as irreflexivity.

Now for a more substantial example of an adequate spatial modal logic: a complete modal logic of planar connection. A modal logic whose theories completely describe connected regions of the plane can be constructed using DL_m plus two planarity axioms (2), (3). Modal formulae such as $\Diamond p$ are to be interpreted as "connects with a region where p is the case". The connection relation is taken to be symmetric, and regions are all distinct. Distinctness can be enforced in extended modal logics by the introduction of *nominals* of the form $p*$, where $p* = p \wedge \neg Dp$ and the "difference operator" D has the semantics $w \models Dp$ iff $\exists v \neq w, v \models p$. The operator $\Diamond_u$ in the planarity axioms is to be read as the universal modality "anywhere". Essentially, the axioms rule out the expression of non-planar connection graphs between regions. Note that isomorphic graphs are dealt with using the iterated modalities $\Diamond^n$.

$$(\neg\Gamma_{3,3}) : \neg \bigwedge_{1\leq i\leq 3} \Diamond_u(p*_{1,i} \wedge \bigwedge_{1\leq k\leq 3} \Diamond^n p*_{2,k}) \qquad (2)$$

$$(\neg\Gamma_5) : \neg \bigwedge_{1\leq i\neq j\leq 5} \Diamond_u(p*_i \wedge \Diamond^n p*_j) \qquad (3)$$

Theorem 2 *The modal logic $DL_m + \neg\Gamma_{3,3} + \neg\Gamma_5$ is a spatially complete logic of connected regions of the plane.*

Proof: Kuratowski's theorem [Kuratowski, 1930] states that all non-planar graphs are of the form of $\Gamma_{3,3}$ or Γ_5. Thus the above axioms suffice to exclude all such graphs. Furthermore, the embedding of any theory of the logic into into R^2 can be carried out by taking the appropriate regions of R^2 to be the graph-theoretic duals of points in the Kripke frame satisfying that theory.

5 FIRST-ORDER SPATIAL LOGICS

In contrast to modal logics, where spatial locations are treated as indices for the evaluation of propositions, first-order spatial logics quantify explicitly over spatial locations and use distinguished predicates to represent relationships between them.

A good example of such a theory is RCC theory [Randell *et al.*, 1992, Gotts *et al.*, 1996, Bennett, 1994, Bennett, 1995]. This approach takes the reflexive and symmetric relation $c(x, y)$, to be interpreted as "the closure of region x is connected to the closure of region y", as primitive. Various other topological and mereological relations between regions are then defined in terms of c, for example "part of" (P), "discrete regions" (DR), "overlaps" (O), and so on. The aim of the theory is to model qualitative spatial relations between regions (of R^3) in first-order logic.

The founders of RCC theory list symmetry and reflexivity of c as the axioms of their theory (see [Gotts, 1994] for additional axioms):

$$\forall x c(x, x)$$
$$\forall x \forall y c(x, y) \rightarrow c(y, x)$$

The basic theory has an extension purporting to capture the notion of convexity. The axioms here are a little more complicated:

$\forall x P(x, conv(x))$
$\forall x P(conv(conv(x)), conv(x))$
$\forall x \forall y \forall z[[P(x, conv(y) \quad \wedge \quad P(y, conv(z))] \quad \rightarrow$
$P(x, conv(z))]$
$\forall x \forall y[[P(x, conv(y) \wedge P(y, conv(x))] \rightarrow O(x, y)]$
$\forall x \forall y[[DR(x, conv(y) \quad \wedge \quad DR(y, conv(x))] \quad \leftrightarrow$
$DR(conv(x), conv(y))]$

Under certain assumptions about its interpretation RCC theory is inadequate in precisely the same sense that von Wright's logic of "near" is. That is, there are consistent sentences of the theory that have no model under the intended interpretation of the spatial primitives. Or, in other words, the axioms merely list *some* properties of the spatial concepts they purport to characterize. Indeed, it is admitted by the RCC researchers that models for their theory are unknown.

Of course, it only makes sense to speak of the adequacy or inadequacy of a spatial logic with respect to an intended interpretation. However, some of the problems with the RCC axiomatization will affect any conceivable classical model.

Furthermore, even if the theory is augmented with axioms guaranteeing the existence of certain regions, another place where things might go wrong involves non-planar graphs. To make things simple, let us suppose for the moment (contrary to the official doctrine of the RCC group) that the regions involved are to be understood as open sets of *two*-dimensional Euclidean space. Let $R_{i,j}$, $(i, j = 1 \text{ to } 3)$ be nine connected regions of the plane. We impose the following constraints on the way that R_i may overlap each other:

1. For $i = 1$ to 3, the three regions $R_{i,1}$, $R_{i,2}$ and $R_{i,3}$ all contain a common (non-empty) region, $T_{1,i}$.
2. For $j = 1$ to 3, the three regions $R_{1,j}$, $R_{2,j}$ and and $R_{3,j}$ all contain a common (non-empty) region, $T_{2,j}$.
3. $R_{i,j}$ and $R_{i',j'}$ do not overlap if $i \neq i'$ and $j \neq j'$.

These constraints can easily be expressed by a formula in the language of RCC given the gloss on $c(x, y)$.

We establish: (i) these constraints cannot be satisfied in R^2; (ii) the formula enforcing these constraints is consistent in RCC. Thus, we have a consistent formula with no model in two-dimensional Euclidean space.

(i) The first two constraints force the existence of six regions $T_{l,m}$ ($l = 1$ to 2, $m = 1$ to 3), such that $T_{1,i}T_{2,i}$ are both contained in $R_{i,j}R_{j,i}$ for all ij. The fact that the $R_{i,j}$ are all connected means that we must be able to draw points $t_{l,m}$ in the regions $T_{l,m}$, and connect these six points with Jordan curves in R^2 so that $t_{1,i}$ is connected to $t_{2,j}$ for all i, j and such that the curve connecting $t_{1,i}$ to $t_{2,j}$ lies wholly within $R_{i,j}$. This graph involving the points $t_{l,m}$ is the graph $\Gamma_{3,3}$, which is well-known to be non-planar. That is, the Jordan curves linking the points must cross at some point. But the third constraint means that these Jordan curves cannot cross, so the constraints cannot be realized in the plane.

(ii) To show that the formula expressing the above constraints is consistent in RCC, we note merely that any graph can be embedded in R^3, so that the above constraints are clearly satisfiable if the regions $R_{i,j}$ are allowed to be three-dimensional. But the axioms of RCC are true of three-dimensional Euclidean space, so the formula expressing the constraints cannot be inconsistent.

Thus we have shown:

Theorem 3 *Let the predicate $c(x, y)$ in the logic RCC have the intended geometrical interpretation "the closure of x has points in common with the closure of y", where x and y are open sets of R^2. Then there are consistent sentences of the logic which have no models with the intended geometrical interpretation.*

We now consider the enhancement of basic RCC theory by means of the convex-hull-forming operator conv(x) [Randell *et al.*, 1992]. This operator is used in the theory to define a set of relations including "inside", "partially inside", and "outside". Suppose that

the basic part of RCC theory were patched up. (We are not certain, at the time of writing, if this is possible.) What of the extension which allows us to express facts about convexity? Again, we show that something needs to be done if the theory is to adequately capture this notion.

Since the axioms for convexity are satisfied by the identity function, this should come as no great surprise. However, we can be much more precise than this. The following theorem gives us an idea of the sort of thing that would be needed if RCC were to be made adequate.

Theorem 4 (Helly's theorem [Eggleston, 1969]) *Let $X_1, \ldots, X_N$, be convex regions in R^n, $N > n$, such that any $n+1$ of the X_i have a non-empty intersection. Then the whole collection of regions has a non-empty intersection.*

Helly's theorem is a non-trivial result when $N > n+1$. In particular, it states that, in the plane, it is impossible to have 4 convex regions each trio of which has a non-empty intersection without all 4 regions having a non-empty intersection. Since this is not the case in three dimensions (imagine the four faces of a tetrahedron), the axioms of RCC theory could not possibly rule the situation out, so there is a consistent formula expressing it, which has no models in R^2. Clearly this result generalizes to higher dimensions. Thus we have deduced (independently of the previous results) a simple result that constrains any search for a logic of convexity.

Theorem 5 *Any axiomatization of* $\mathrm{conv}(x)$ *in the language of RCC which is adequate for n-dimensional Euclidean space must be incorrect for $n+1$-dimensional Euclidean space.*

6 CONCLUSION

We have provided a rigorous semantical adequacy criterion for spatial logics. This venture has drawn out problems and issues for RCC theory in particular and spatial logics in general. Even a simple modal "spatial" logic [von Wright, 1979, von Wright, 1984] is shown to fail to meet the adequacy criterion, although it is shown that some quite trivial positional notions (which are informally thought of as spatial) can be captured in modal systems. We have shown that the RCC theories of "space" can express consistent theories which do not have spatial models. Furthermore, a modal spatial logic is provided, and shown to meet the adequacy criterion. The results herein highlight the importance, for a wide variety of research programmes, of a deeper understanding of the formal semantics of spatial logics.

Acknowledgements

The author is indebted to Ian Pratt, and to Jeff Paris, for pointing out the argument used in theorem 3. This research is supported by the Leverhulme Trust.

References

[Allen, 1981] J. F. Allen. An interval-based representation of temporal knowledge. In *7th IJCAI*, 1981.

[Asher and Vieu, 1995] Nicholas Asher and Laure Vieu. Toward a Geometry of Common Sense: a semantics and a complete axiomatization of mereotopology. In *IJCAI '95*, 1995.

[Balbiani *et al.*, 1995] Philippe Balbiani, Luis Farinas del Cerro, Tinko Tinchev, and Dimiter Vakarelov. An Intensional Model of Space. In *Time, Space, and Movement*, pages 113–121, Toulouse, 1995. Groupe LRC Toulouse.

[Bennett, 1994] Brandon Bennett. Spatial Reasoning with Propositional Logics. In *4th International Conference on Knowledge Representation and Reasoning*. Morgan Kaufmann, 1994.

[Bennett, 1995] Brandon Bennett. Modal Logics for Qualitative Spatial Reasoning. *Bulletin of the IGPL*, 3, 1995.

[Biacino and Gerla, 1991] Loredana Biacino and Giangiacomo Gerla. Connection Structures. *Notre Dame Journal of Formal Logic*, 32(2):242 – 247, 1991.

[Clarke, 1981] B. L. Clarke. A calculus of individuals based on 'connection'. *Notre Dame Journal of Formal Logic*, 23(3):204 – 218, 1981.

[Clarke, 1985] B. L. Clarke. Individuals and Points. *Notre Dame Journal of Formal Logic*, 26(1):61 – 75, 1985.

[de Rijke, 1992] Maarten de Rijke. The Modal Logic of Inequality. *Notre Dame Journal of Formal Logic*, 57(2):566 – 584, 1992.

[Eggleston, 1969] H. G. Eggleston. *Convexity*. Cambridge University Press, Cambridge, 1969.

[Frank and Campari, 1993] Andrew Frank and Irene Campari, editors. *COSIT '93: Spatial Information Theory: a theoretical basis for GIS*, volume 716 of *Lecture Notes in Computer Science*. Springer Verlag, Berlin, 1993.

[Frank and Kuhn, 1995] Andrew Frank and Werner Kuhn, editors. *COSIT '95: Spatial Information Theory: a theoretical basis for GIS*, volume 988 of *Lecture Notes in Computer Science*. Springer Verlag, Berlin, 1995.

[Georgatos, 1994] Konstantinos Georgatos. Knowledge Theoretic Properties of Topological Spaces. In Michael Masuch and Laszlo Polos, editors, *Knowledge Representation and Reasoning Under Uncertainty: Logic at work*, volume 808 of *LNAI*, pages 147–160. Springer-Verlag, Berlin, 1994.

[Gotts *et al.*, 1996] Nicholas Gotts, John Gooday, and Anthony Cohn. A connection based approach to commonsense topological description and reasoning. *Monist*, 79(1):51–75, 1996.

[Gotts, 1994] Nicholas Mark Gotts. How far can we 'C'? Defining a 'Doughnut' using connection alone. In *4th International Conference on Knowledge Representation and Reasoning*. Morgan Kaufmann, 1994.

[Hernández, 1994] Daniel Hernández. *Qualitative Representation of Spatial Knowledge*, volume 804 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin, 1994.

[Jeansoulin and Mathieu, 1995] Robert Jeansoulin and Christophe Mathieu. Revisable Spatial Knowledge by means of a Spatial Modal Logic. In *Joint European Conference on GIS*, The Hague, Netherlands, 1995.

[Kuratowski, 1930] G. Kuratowski. Sur le probleme des courbes gauches en topologie. *Fund. Math.*, 15-16:271, 1930.

[Lemon and Pratt, 1996] Oliver Lemon and Ian Pratt. Putting Channels on the Map: imperfect information flow in a formal semantics of (geo)graphical information systems. In *Information Theoretic Approaches to Logic, Language, and Computation*, Proceedings of the 2nd conference on Information Theoretic Approaches to Logic, Language, and Computation (ITALLC), London Guildhall University, 1996. Department of Psychology.

[Lemon, 1996] Oliver Lemon. Theories of Representation. In *Logica '96 international symposium*, Prague (forthcoming), 1996. Filosofia Academic Publishing.

[Mark and Frank, 1991] David Mark and Andrew Frank, editors. *Cognitive and Linguistic Aspects of Geographic Space*, volume 63 of *NATO ASI Series D*. Kluwer, Dordrecht/Boston/London, 1991.

[Moss and Parikh, 1992] Lawrence Moss and Rohit Parikh. Topological Reasoning and the Logic of Knowledge. In Y. Moses, editor, *Theoretical Aspects of Reasoning about Knowledge: 4th conference (TARK '92)*, pages 95–105, 1992.

[Nebel, 1995] Bernhard Nebel. Computational Properties of Qualitative Spatial Reasoning: First Results. In Ipke Wachsmuth, Claus-Rainer Rollinger, and Wilfred Brauer, editors, *KI-95: Advances in Artificial Intelligence*. Springer-Verlag, Berlin, 1995. 19th German Conference on Artificial Intelligence.

[Pratt, 1993] Ian Pratt. Map Semantics. In Andrew Frank and Irene Campari, editors, *Spatial Information Theory: a theoretical basis for GIS*, volume 716 of *Lecture Notes in Computer Science*, pages 77 – 91. Springer Verlag, Berlin, 1993.

[Randell *et al.*, 1992] David Randell, Zhan Cui, and Anthony Cohn. A Spatial Logic based on Regions and Connection. In *3rd International Conference on Knowledge Representation and Reasoning*. Morgan Kaufmann, 1992.

[Rescher and Garson, 1968] Nicholas Rescher and James Garson. Topological Logic. *Journal of Symbolic Logic*, 33:537–548, 1968.

[Rescher, 1979] Nicholas Rescher. Reply to von Wright. In E. Sosa, editor, *The Philosophy of Nicholas Rescher: Discussion and Replies*, pages 74–75. D. Reidel, Dordrecht, Holland, 1979.

[Tarski, 1956] Alfred Tarski. *Logic, Semantics, Metamathematics*. Claredon Press, Oxford, 1956.

[van Benthem, 1984] Johan van Benthem. Correspondence Theory. In Dov Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 3. Reidel, Dordrecht, 1984.

[von Wright, 1979] Georg Henrik von Wright. A Modal Logic of Place. In E. Sosa, editor, *The Philosophy of Nicholas Rescher: Discussion and Replies*, pages 65–73. D. Reidel, Dordrecht, Holland, 1979.

[von Wright, 1984] Georg Henrik von Wright. *Philosophical Logic*. Basil Blackwell, Oxford, 1984. (Philosophical Papers volume II).

[Wagon, 1993] Stan Wagon. *The Banach-Tarski Paradox*. Cambridge University Press, Cambridge, 1993.

[Whitehead, 1929] Alfred North Whitehead. *Process and Reality*. The MacMillan Company, New York, 1929.

[Worboys, 1991] Michael Worboys. The Role of Modal Logics in the Description of a Geographical Information System. In David Mark and Andrew Frank, editors, *Cognitive and Linguistic Aspects of Geographic Space*, volume 63 of *NATO ASI Series D*. Kluwer, Dordrecht/Boston/London, 1991.

A Pointless Theory of Space Based On Strong Connection and Congruence

Stefano Borgo, Nicola Guarino, and Claudio Masolo
National Research Council,
LADSEB-CNR
Corso Stati Uniti, 4
I-35127 Padova, Italy
{borgo, guarino, masolo}@ladseb.pd.cnr.it

Abstract

We present a logical theory of space where only tridimensional regions are assumed in the domain. Three distinct primitives are used to describe their mereological, topological and morphological properties: mereology is described by a parthood relation satisfying the axioms of Closed Extensional Mereology; topology is described by means of a "simple region" predicate, by which a relation of "strong connection" between regions having at least a surface in common is defined; morphology is described by means of a "congruence" primitive, whose axioms exploit Tarski's analogy between points and spheres.

1 INTRODUCTION

Various logical theories aimed at the representation of commonsense spatial knowledge have been proposed in the AI community in recent years. In the spirit of (Hayes 1985a), the goal has been that of establishing the logical basis of a "geometry of commonsense", intended to be used for tasks as disparate as robot navigation or natural language understanding. Besides specific proposals focused on particular aspects, like (Hayes 1985b, Davis 1993, Shanahan 1995), the most general frameworks have been perhaps those based on a combination of mereological and topological notions, which appeared in AI after the publication of Clarke's mereotopological theory (Clarke 1981). Among these, particular relevance for AI purposes has the so-called RCC theory (Randell and Cohn 1992, Gotts 1994, Bennett 1995, Gotts *et al.* 1996), which has been recently joined by other proposals originating in the area of philosophy and linguistics (Aurnague and Vieu 1993, Asher and Vieu 1995, Eschenbach and Heydrich 1995, Casati and Varzi 1996, Smith 1996). These approaches differ in the primitives adopted and in the ontological assumptions about the domain. They all have in common however the use of the tools of so-called "formal ontology" (Guarino 1995) for the representation of commonsense reality: specifically, mereology and topology. Readers can refer to (Simons 1987) for a general overview of mereology, and to (Varzi 1994, Varzi 1996b) for a systematic account of the subtle relations between mereology and topology.

We can distinguish four main aspects which are useful to compare and evaluate ontological theories of space, namely: i) the assumptions related to the ontological status of spatial entities (conceived of as existing either only on its own or rather only relative to physical objects); ii) the presence/absence of lower dimension entities like *boundaries* (surfaces, lines or points) in the domain; iii) the primitives used to express the relevant relations; iv) the degree of characterization of the intended models. In this paper, we are presenting a logical theory whose main contributions are discussed and motivated in the light of these points.

In section 2 we address the first two points, discussing our basic ontological assumptions about space. We adopt the classical conception of absolute space, excluding lower dimension entities from the domain; our individuals are therefore three-dimensional regions.

The discussion of points iii and iv represents the body of the paper. We assume three distinct levels of description of space: the *mereological level*, the *topological level*, and the *morphological level*. A specific primitive is adopted for each of these levels.

In fact, since our domain is limited to spatial entities only, we could use the connection ('C') primitive to define parthood (as done by Clarke), eliminating of a primitive. We prefer however to keep the mereological and topological levels separate because of reasons of cognitive clarity and immediateness, adopting a general parthood primitive supplemented with a specific topological primitive. Our choice differs therefore from the approaches inspired by Clarke's work, since we don't take connection ('C') as a primitive: rather, we introduce a unary *simple region* ('SR') primitive, where a simple region (shortly, *s-region*), is a region "all in one piece", i.e. a *strongly* self-connected region, such that any two halves of it always

have a surface in common (*strong connection* or *surface connection*, briefly *s-connection*). In other words, an s-region, in everyday intuition, can be occupied by a single thing. The reasons of this choice come from the desire to make possible a simple and natural interpretation of the very notion of (external) connection, which is far from being totally clear, especially for what concerns the distinctions between point-, line- and surface-connection.

A distinguishing feature of the theory presented here is the emphasis given to the morphological level. We take the *congruence relation* between regions as a primitive, proposing an axiomatization which is constructed upon the reconstruction of Euclidean geometry in terms of spheres developed by Tarski (Tarski 1956). Spheres, in turn, are defined in terms of parthood, s-regions and congruence. The resulting theory turns out to be extremely powerful, allowing us to rigorously distinguish between point-, line-, and surface-connection, and to define notions such as convexity and granularity, which we plan to exploit in a research project aimed at the logical modelling of mechanical assemblies (Borgo *et al.* 1996).

For each of the three levels introduced above, we propose an axiomatization aimed at achieving a good characterization of our intended models. This last point marks a major feature of our approach: our theory is an ontological theory, in the sense discussed in (Guarino and Giaretta 1995). It is a *rich* theory in terms of axioms and definitions (in the spirit of (Hayes 1985)), since its main purpose is to *convey meaning*, in such a way as to characterize more precisely the ontological assumptions underlying the primitives adopted, restricting their possible interpretations. The high expressiveness of the language (full first order logic) makes the theory not suitable for direct implementation in a reasoning system, while making it possible however to adopt its primitives and definitions as the basis of a logical vocabulary suitable to be *shared* among different applications (Guarino *et al.* 1994).

The rest of this paper develops as follows. In section 2 we introduce in more detail our basic ontological assumptions; sections 3, 4 and 5 are devoted to the presentation of our mereological, topological and morphological axioms, respectively; finally, in section 6 we make an overall assessment of the theory with respect to the current literature, and we briefly discuss its potential applications.

2 BASIC ONTOLOGICAL ASSUMPTIONS

We have adopted here the classical Newtonian conception of space, according to which space exists independently of the entities that can occupy it. In this view, space is like a container where physical objects can be located and can move. In other words, space is intended as a *substrate*, since its existence is a necessary condition for physical objects to exist (Borgo *et al.* 1996). Other proposals more oriented towards natural language applications, like (Aurnague and Vieu 1993), adopt the Leibnizian conception, assuming space to be strictly dependent on the relations holding between physical objects. In this second view, it is not necessary to admit the existence of spatial entities like regions, and, in any case, the existence of such entities depends directly on the objects themselves. Although both of these claims are tenable, the first position seems to be much easier for practical applications. For instance, in the absolute conception, the movement of an object could be described quite simply as the sequence of regions where the object is located, whereas otherwise we would have to consider the changes of the *relevant* relations between our object and the other objects in the system.

A further crucial assumption is related to the exclusion of spatial entities like surfaces, lines, and points from our domain. They are basic ingredients of ordinary geometry, but their ontological status in our everyday interaction with the world is debatable. In fact, our experience of space is strictly related to physical three-dimensional objects (our human body is one of them). We move these objects around to perceive space and to recognise the "outside" world. From this naive position, points are not considered as inhabitants of this world, but rather results of complex abstract reasonings, so they could be defined, if really necessary, as higher-order entities. In this spirit, various proposals for an axiomatization of *pointless geometry* have been made in the past (Lobachevskij 1835, Whitehead 1929, Grzegorczyk 1960) (see (Gerla 1994) for an overview). Approaches based on Clarke's axiomatization, like (Asher and Vieu 1995, Gotts *et al.* 1996), don't admit points in their domain[1]. The RCC theory moves further, getting rid of the topological distinction between open and closed regions to avoid a number of problems related to the boundaries of physical objects. We have adopted this latter choice, assuming regular three-dimensional regions in a 3D Euclidean space as elements of our domain. The peculiar property exhibited by these regions is that they can be occupied by concrete physical objects. Indeed, although the domain assumed in the formalization below is limited to spatial regions only, all the ontological choices made in this paper are motivated by a more general view where space, matter and physical objects are considered (Borgo *et al.* 1996).

3 MEREOLOGICAL LEVEL

Let us present the axioms assumed for our primitive parthood relation, represented by means of the binary predicate 'P'. It is important to stress that our domain is limited here to spatial regions; in this restricted domain, parthood can be taken as equivalent to spatial inclusion, since the various problems bound to the ontological nature of the entities involved disappear: for instance, it makes no sense to distinguish an object in a hole (which is not

[1] See however (Varzi 1996a) and (Smith 1996) for a different strategy, where the peculiar ontological status of *boundaries* is defended.

part of the hole) from the region it occupies (which is part of the region occupied by the hole) (Casati and Varzi 1995).

We have adopted in the following a standard first-order language with identity. Throughout the paper, free variables appearing in formulas are intended to be universally quantified. The following definitions are built upon the parthood relation[1]:

D1. $PPxy =_{df} Pxy \wedge \neg x=y$ *(Proper part)*

D2. $Oxy =_{df} \exists z(Pzx \wedge Pzy)$ *(Overlap)*

D3. $POxy =_{df} Oxy \wedge \neg Pxy \wedge \neg Pyx$ *(Proper overlap)*

D4. $x+y =_{df} \iota z \forall w(Owz \leftrightarrow (Owx \vee Owy))$ *(Sum)*

D5. $x-y =_{df} \iota z \forall w(Pwz \leftrightarrow (Pwx \wedge \neg Owy))$ *(Difference)*

D6. $x \times y =_{df} \iota z \forall w(Pwz \leftrightarrow (Pwx \wedge Pwy))$ *(Product)*

The ι operator appearing above is defined contextually *à la* Russel:

D7. $\psi[\iota x \phi] =_{df} \exists y(\forall x(\phi \leftrightarrow x=y) \wedge \psi[y])$

The following axioms – equivalent to those of Closed Extensional Mereology (Simons 1987, Varzi 1996b) – are introduced:

A1. Pxx

A2. $Pxy \wedge Pyx \rightarrow x=y$

A3. $Pxy \wedge Pyz \rightarrow Pxz$

A4. $\exists z(z=x+y)$

A5. $\neg Pxy \rightarrow \exists z(z=x-y)$

Notice that the so-called "fusion axiom" (allowing for infinite sums) typical of General Extensional Mereology is not assumed, since it would lead to inconsistencies in our mereo-topological framework (see below). Some simple theorems deriving from the axioms above are the following:

T1. $\neg Pxy \rightarrow \exists z(Pzx \wedge \neg Ozy)$ (A1; A5; D5)
(strong remainder principle)

T2. $Oxy \rightarrow \exists z(z=x \times y)$ (A4; A5; D6)
(existence of product)

4 TOPOLOGICAL LEVEL

As mentioned in the introduction, we have decided to adopt a unary primitive 'SR' instead of 'C'. To make clear this choice it is important to analize the relation between mereology and topology a little more. In the past, various attempts have been made to define topological notions in terms of parthood. In fact, the overlap relation defined by D2:

$Oxy =_{df} \exists z(Pzx \wedge Pzy)$

seems to resemble a notion akin to that of connection. The crucial point here is the ontological nature of the elements involved in the above definition. As pointed out in (Eschenbach and Heydrich 1995), if we have both points and regions in the domain we can restrict the arguments of the "overlap" relation to hold only for regions, by means of a primitive predicate 'R', which selects (regular) regions. Topological connection can be therefore defined as follows:

$Cxy =_{df} Rx \wedge Ry \wedge Oxy$ *(connection)*

In the case of external connection, this means that x and y have a part in common (i.e., they overlap), but this part is not a region (assuming regions as three-dimensional, it may be a point, a line, or a surface).

Notice that the primitive 'R' above isolates some entities in the domain which have a peculiar "topological" meaning. In our case, since the domain is *already* limited to regions, the definition above would collapse to ordinary overlap. We must then resort to some other strategy. An attempt in this sense has been made by Whitehead (Whitehead 1929), who tried to define connection as follows, by means of mereology only:

$Cxy =_{df} \exists z(Ozx \wedge Ozy \wedge \forall w(Pwz \rightarrow (Owx \vee Owy)))$

As shown in (Varzi 1994), it is easy to see however that the above definition fails if we allow z to be a disconnected region (Figure 1).

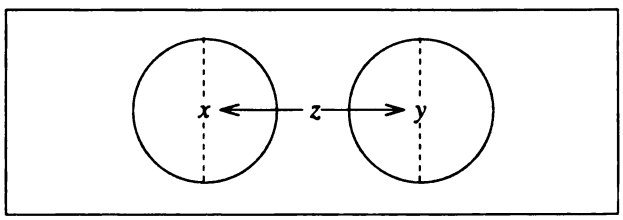


Figure 1. Whitehead's problem: x and y are not connected unless the overlaying piece z is itself connected. From (Casati and Varzi 1995).

In other words, we face here the main problem of mereology, namely the impossibility of characterizing the notion of a *whole*. The move we have made to solve this problem is simply the introduction of this notion at the topological level by means of the primitive 'SR' which denotes a *self-connected* regular region. This will allow us to define "connection" in terms of 'P' and 'SR' by means of a simple specialization of Whitehead's definition:

$Cxy =_{df} \exists z(SRz \wedge Ozx \wedge Ozy \wedge \forall w(Pwz \rightarrow (Owx \vee Owy)))$

We see therefore that an extra primitive besides parthood is necessary to define connection. The *meaning* of the connection relation, however, remains still obscure unless the 'SR' primitive is suitably characterized. Our choice

[1] Throughout the paper, we mark definitions with 'D', axioms with 'A' and theorems with 'T'

has been to interpret 'SR' in a strong sense, as denoting s-regions rather than generic self-connected regions. Under this interpretation, the definition above refers to *strong connection* (*s-connection*). In our intuition, such a notion is bound to that of *physical connection.*. For an example of a *non* s-connected region, think of two spheres (say, two apples) touching each other in a point: no physical object (like a worm for instance, however thin) can pass from one to the other without exposing itself to the outside space (Figure 2). We shall say in this case that the two spheres are connected but not s-connected. Later, with the help of morphology, we will be able to distinguish also between point- and line-connection (*p-connection* and *l-connection*).

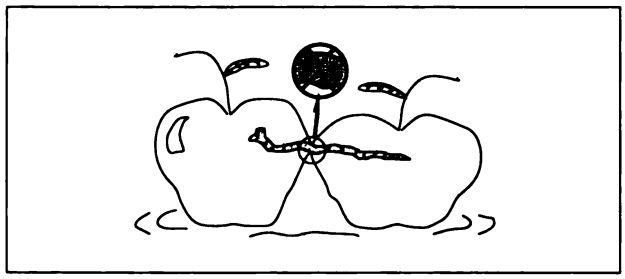


Figure 2. The regions occupied by the two apples are not s-connected. Their sum is not an s-region.

After this discussion of our *desiderata*, let us introduce the axioms and definitions which characterize our topological level, together with some interesting theorems.

Definitions:

D8. $IPxy =_{df} PPxy \wedge \forall z((SRz \wedge POzx) \rightarrow Oz(y\text{-}x))$
(interior part)

D9. $MCPxy =_{df} Pxy \wedge SRx \wedge \neg\exists z(Pzy \wedge SRz \wedge PPxz)$
(maximally connected part)

D10. $SCxy =_{df} \exists uv(Pux \wedge Pvy \wedge SR(u+v))$
(strong connection)

Axioms:

A6. $(SRx \wedge x{=}y{+}z) \rightarrow \exists u(SRu \wedge Ouy \wedge Ouz \wedge IPux)$

A7. $\exists y\ MCPyx$

A8. $\exists y\ (SRy \wedge IPxy)$

Definition D8 deserves some comments. Notice first that, in our interpretation, the relation 'IP' turns out to be different from the usual relation 'NTPP' (non-tangential proper part), since a region x being l- or p-connected with a region external to y must be considered as an interior part of y (Figure 3). A further observation is related to the possibility of allowing the infinite mereological sum of the interior parts of a given region. This would lead to inconsistencies, since such a sum would coincide with the maximal interior part, while no maximal interior part can be admitted by T10 (see below). For this reason we have excluded infinite sums from our mereological framework. A similar problem has been noticed in (Randell and Cohn 1992).

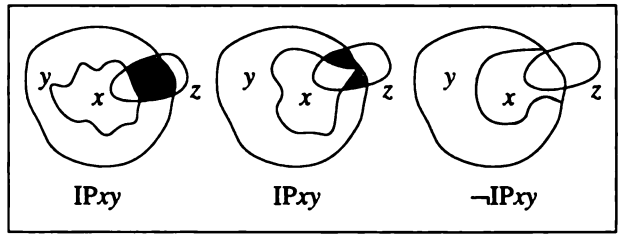


Figure 3. No external region can be *strongly* connected to interior parts.

Definition D9 introduces the notion of maximally connected part, and definition D10 is a compact rewriting of Whitehad's amended definition reported above.

Axiom A6 refines the idea of strong connection between two halves of a s-region by adding the further requirement that the "connecting" region be an interior part of the resulting sum; in this way, it excludes regions which are not manifolds, like the sum of two tangent spheres, to be considered as a s-region. To see this, suppose *per absurdum* that the sum x of two p-connected regions y and z is a simple region (Figure 4). According to A6, we must find a "connecting" s-region u overlapping both y and z, which is at the same time an interior part of x. Now, the only possible candidates for u must be regions p-connected in the same point which connects y and z, but these can't be interior parts of x since there will be other regions (like w) external to x touching u in the same point. An analogous argument applies to the case of l-connected regions.

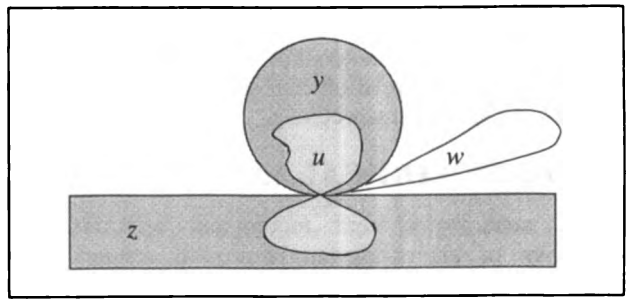


Figure 4. Why $y{+}z$ cannot be an s-region.

Axioms A7 and A8 further determine the structure of our spatial domain. Through A7, every region has some

maximally connected part; A8 states that any region is part of some simple region, and therefore the notions of universe and complement are not defined. Among other things, this axiom prevents 'SR' from being interpreted as a "local" property, like for example 'being red'.

Let us now present some theorems which follow from our axiomatization:

T3. $IPxy \wedge Pyz \rightarrow IPxz$ (A3; D8)

T4. $Pxy \wedge IPyz \rightarrow IPxz$ (A3; D8)

T5. $IPxy \wedge IPxz \rightarrow IPx(y \times z)$ (D6; D8)

T6. $IPxz \wedge IPyz \rightarrow IP(x+y)z$ (D4; D8)

T7. $SRx \leftrightarrow MCPxx$ (D9)

T8. $\neg SRx \rightarrow \exists y(MCPyx \wedge PPyx)$ (A7; T7; D9)

T9. $\exists y(SRy \wedge IPyx)$ (A6; T3; T8; D9)

T10. $IPxy \rightarrow \exists z(IPzy \wedge PPxz)$ (A4; A5; T3; T6; T9)

T11. $SRx \rightarrow \forall yz(y+z=x \rightarrow SCyz)$ (A6; D10)

T12. $\exists y(SCxy \wedge \neg Oxy)$ (A5; A8)

T13. $SCxx$ (A8; D10)

T14. $SCxy \rightarrow SCyx$ (D10)

T15. $Pxy \leftrightarrow \forall z(SCzx \rightarrow SCzy)$ (A3; A5; T1; T3; T9)

T16. $x=y \leftrightarrow \forall z(SCzx \leftrightarrow SCzy)$ (A2; T15)

Theorems T3-T6 establish some desirable properties of the relation 'IP' with respect to mereology, which are actually taken as axioms in (Smith 1996), where the 'IP' relation is adopted as a primitive; T7 shows that an s-region is maximally self-connected, while T8 shows that given a non-simple region it has a maximally connected proper part. T9 shows that our theory is not atomic, and that any region has an interior part which is an s-region. T10 shows that our space is dense, in the sense that, given a region x internal to y, there always exists another region z properly including x and still internal to y[1]. T11 shows that two halves of an s-region are always s-connected, while T12 shows that our notion of strong connection is actually different from overlap. Finally, T13-T14 show the reflexivity and the symmetry of s-connection, while T15-T16 establish the relationship of s-connection with parthood and identity, respectively.

5 MORPHOLOGICAL LEVEL

Besides mereological and topological relations, the possibility of expressing morphological features is a crucial aspect for any commonsense theory of space. A "convex hull" primitive has been used with some success within the RCC theory, but it seems too weak for our purpose. A ternary alignment relation has been used in (Aurnague and Vieu 1993), but it commits to a (higher-order) notion of point. We have opted here for a different morphological primitive, the most intuitive we can think of: the *congruence relation* between regions, designated by 'CG', meaning that CGxy holds if x and y have the same shape and size. In the case of classical geometry based on points, segments and angles, this relation was first axiomatized in (Hilbert 1902), with various simplifications thereafter. In our restricted domain, a "direct" axiomatization of the notion of congruence between regions seemed to be quite complicated to us, since we are not aware of any similar approach. We have chosen therefore to take advantage of Hilbert's work, by exploiting a correpondence between points and spheres. This correspondence has been brilliantly pursued in (Tarski 1956), where a (second order) axiomatic theory taking spheres and parts as primitives has been shown as equivalent to classical geometry. Our strategy to axiomatize 'CG' is the following:

1. define a sphere in terms of 'P', 'SR' and 'CG';

2. adopt Tarski's definitions related to spheres;

3. define a notion of alignment for three spheres;

4. formulate axioms for the congruence between binary sums of spheres (s-segments) by mimicking the standard axioms for congruence between segments, exploiting the analogy between points and spheres and the above definition of alignement for spheres

5. constrain the congruence between arbitrary regions in terms of congruence between binary sums of spheres.

6. add further axioms in order to constrain the congruence between spheres, and introduce suitable existential assumptions.

The result will be a *first order* theory of congruence between regions: with respect to Tarski's work, we *reconstruct* the congruence axioms in terms of spheres, rather than establishing a formal connection between points and classes of concentrical spheres (going therefore to second order). Moreover, we do not take spheres as a primitive, since we define them in terms of general mereological, topological, and morphological primitives.

5.1 DEFINING SPHERES

The crucial step in our axiomatization of congruence is the definition of a sphere, that makes it possible to embed Tarski's mereo-morphological theory within our framework:

D11. $SPHx =_{df} SRx \wedge \forall y(CGxy \wedge POxy \rightarrow SR(x-y))$

A region x is a spere if and only if it is a simple region and it cannot be disconnected by another congruent sphere. It is easy to see that only spherical regions satisfy the above definition, which reflects the intrinsic symmetry of spheres (Figure 5), provided that enough regions congruent to the one given exist.

[1] This is a weak notion of density. We cannot prove a stronger notion as: $IPxy \rightarrow \exists z(IPzy \wedge IPxz)$

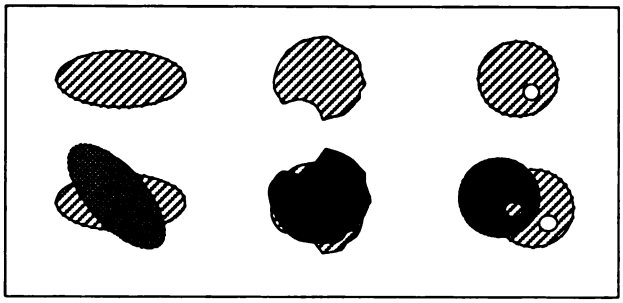


Figure 5. These regions are not spheres.

5.2 TARSKI'S DEFINITIONS

Let us now introduce Tarski's definitions regarding spheres. For the sake of conciseness, we assume that all variables appearing in D12-D21 below are restricted to the class of spheres.

D12. $ETxy =_{df} \neg Oxy \wedge$
$((\neg Ouy \wedge \neg Ovy \wedge Pxu \wedge Pxv) \rightarrow (Puv \vee Pvu))$
(x is externally tangent to y)

D13. $ITxy =_{df} PPxy \wedge$
$((Puy \wedge Pvy \wedge Pxu \wedge Pxv) \rightarrow (Puv \vee Pvu))$
(x is internally tangent to y)

D14. $EDxyz =_{df} ETxz \wedge ETyz \wedge$
$(\neg Ouz \wedge \neg Ovz \wedge Pxu \wedge Pyv \rightarrow \neg Ouv)$
(x and y are externally diametrical wrt z)

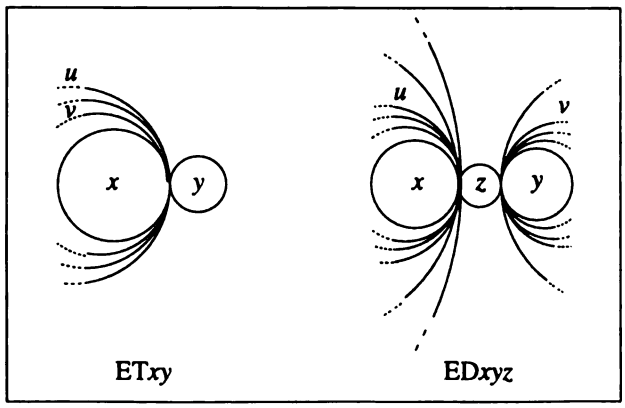


Figure 6. Externally tangent and externally diametrical spheres (D12, D14).

D15. $IDxyz =_{df} ITxz \wedge ITyz \wedge$
$(\neg Ouz \wedge \neg Ovz \wedge ETxu \wedge ETyv \rightarrow \neg Ouv)$
(x and y are internally diametrical wrt z)

D16. $CNCxy =_{df} x = y \vee$
$(PPxy \wedge (EDuvx \wedge ITuy \wedge ITvy \rightarrow IDuvy)) \vee$
$(PPyx \wedge (EDuvy \wedge ITux \wedge ITvx \rightarrow IDuvx))$
(x is concentric with y)

5.3 ALIGNMENT OF SPHERES

We can now easily exploit Tarski's definitions to define the notion of alignment among spheres, and to establish notions analogous to those of segments and triangles in terms of spheres (*s-segments* and *s-triangles*):

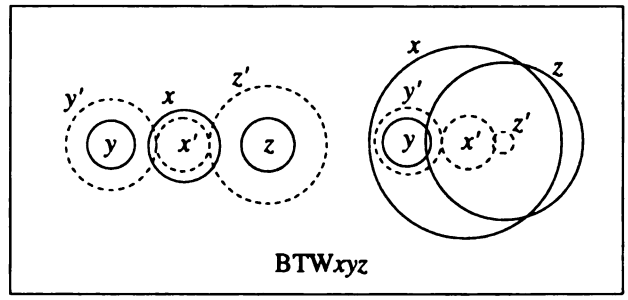


Figure 7. *x* is *between* *y* and *z* (D17).

D17. $BTWxyz =_{df} \exists x'y'z'(CNCxx' \wedge CNCyy' \wedge$
$CNCzz' \wedge EDy'z'x')$
(x is between y and z)

D18. $LINxyz =_{df} BTWxyz \vee BTWyxz \vee BTWzxy$
(x is aligned wrt y and z)

D19. $SSDxyz =_{df} BTWxyz \vee BTWyxz \vee CNCxy$
(x and y are on the same side wrt z)

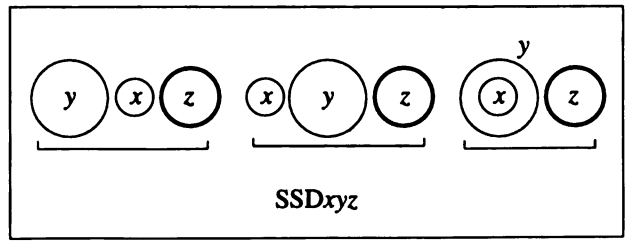


Figure 8. *x* is *on the same side* of *y* with respect to *z* (D19)

D20. $SEGxy =_{df} \neg CNCxy$ *(x and y form an s-segment)*

D21. $TRIxyz =_{df} \neg CNCxy \wedge \neg CNCyz \wedge \neg CNCxz \wedge$
$\neg LINxyz$
(x, y and z form an s-triangle)

We restrict now the notions of s-segment and s-triangle in order to exclude the case where one of the spheres is part of some other one.

D22. $PNP2xy =_{df} \neg Pxy \wedge \neg Pyx$

D23. $PNP3xyz =_{df} PNP2xy \wedge PNP2xz \wedge PNP2yz$
(pairwise not part)

D24. $PBTWxyz =_{df} BTWxyz \wedge PNP3xyz$ *(proper between)*

D25. $PSEGxy =_{df} SEGxy \wedge PNP2xy$ *(proper s-segment)*

D26. $PTRIxyz =_{df} TRIxyz \wedge PNP3xyz$ *(proper s-triangle)*

5.4 RECONSTRUCTING STANDARD AXIOMS FOR CONGRUENCE

We can now introduce the proper axioms of congruence, modifying the formulation presented in (Coxeter 1989), which is a variation of Hilbert's system, by exploiting the parallelism between points and spheres.

A9. $CGxy \wedge CGzy \rightarrow CGxz$

A10. $PSEGxy \wedge SEGzw \rightarrow \exists!x'y'(CG(x+y)(x'+y') \wedge CNCzx' \wedge CGxx' \wedge SSDwy'x')$
(Transportability of s-segments)

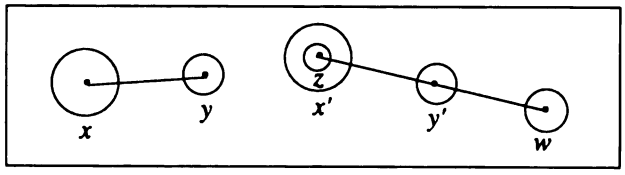


Figure 9. Transportability of s-segments (A10). Given a proper s-segment xy and an arbitrary segment zw, there exists a unique s-segment $x'y'$ congruent to xy such that x' is concentrical to z and congruent to x, while y' is on the same side of w with respect to x'.

A11. $PBTWyxz \wedge BTWy'x'z' \wedge CGxx' \wedge CG(x+y)(x'+y') \wedge CG(y+z)(y'+z') \rightarrow CG(x+z)(x'+z')$
(Congruence of s-segments)

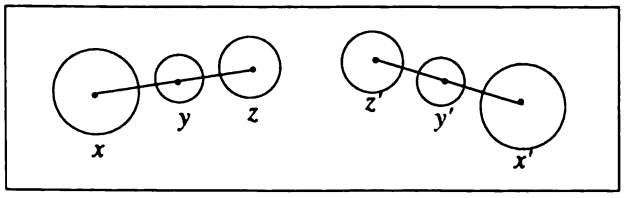


Figure 10. Congruence of s-segments (A11). If xy and yz are respectively congruent to $x'y'$ and $y'z'$, then xz is congruent to $x'z'$ provided that xyz and $x'y'z'$ are aligned.

A12. $PTRIxyz \wedge TRIx'y'z' \wedge PBTWyxv \wedge BTWy'x'v' \wedge CGxx' \wedge CGyy' \wedge CGzz' \wedge CGvv' \wedge CG(x+y)(x'+y') \wedge CG(x+z)(x'+z') \wedge CG(y+z)(y'+z') \wedge CG(x+v)(x'+v') \rightarrow CG(z+v)(z'+v')$
(Congruence in s-triangles)

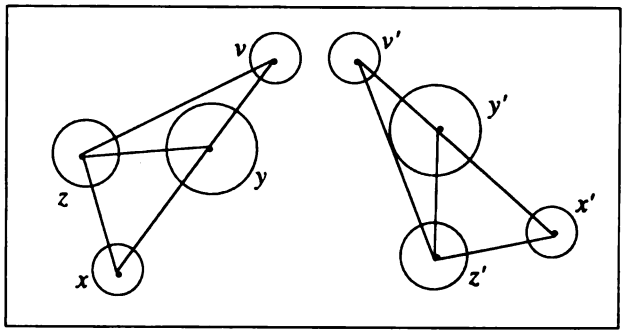


Figure 11. Congruence in s-triangles (A12). The s-segment vz is congruent to $v'z'$ if the s-triangle xyz is congruent to the s-triangle $x'y'z'$ and xv is congruent to $x'v'$.

The axiom A9, together with suitable existential assumptions, implies that 'CG' is an equivalence relation. Axioms A10-A12 are described by Figures 9-11.

The axioms above constrain the congruence between binary sums of spheres (s-segments).

5.5 CONGRUENCE BETWEEN ARBITRARY REGIONS

In the following, we first establish a notion of congruence between (suitable) sums of spheres, and then we use such a notion to constrain the congruence between arbitrary regions.

When considering the congruence between sums of spheres, we could follow the reasoning used by Hilbert in order to establish the congruence between sums of points (figures); to do that, we need to establish a suitable bijection between the two sums of spheres such that the corresponding s-segments are congruent. In order to avoid looking at all possible bijections (which would require a second-order axiomatization), we define an *ad hoc* congruence relation holding for *scalene* sums of spheres (sums of not s-connected spheres such that all of them are of different sizes). This means that, with respect to the size of the spheres, only one bijection exists. Once we have congruence between scalene sums of spheres, we use it to constrain congruence between arbitrary regions.

D27. $\Sigma SSx =_{df} \forall y(MCPyx \rightarrow (SPHy \wedge \neg\exists z\ (MCPzx \wedge \neg\ y{=}z \wedge CGzy)))$ *(scalene sum of spheres)*

D28. $\Sigma CGxy =_{df} \Sigma SSx \wedge \Sigma SSy \wedge$
$\forall z(MCPzx \rightarrow \exists w(MCPwy \wedge CGzw)) \wedge$
$\forall z(MCPzy \rightarrow \exists w(MCPwx \wedge CGzw)) \wedge$
$\forall uv(MCPux \wedge MCPvx \wedge SEGuv) \rightarrow$
$\exists u'v'(MCPu'y \wedge MCPv'y \wedge CG(u+v)(u'+v'))$
(congruent scalene sums of spheres)

A13. $CGxy \leftrightarrow$
$(\forall z(\Sigma SSz \wedge Pzx \rightarrow \exists w(\Sigma CGzw \wedge Pwy)) \wedge$
$\forall z(\Sigma SSz \wedge Pzy \rightarrow \exists w(\Sigma CGzw \wedge Pwx)))$
(congruence of arbitrary regions)

Notice that the second and third lines in D28 constrain the two sums to have the same number of spheres. By means of A13 we can conclude that $\Sigma CGxy \rightarrow CGxy$.

5.6 FURTHER CONSTRAINTS

We need now to add some further axioms constraining the congruence between single spheres. Hilbert's axioms are of no help here, since he obviously assumes that all points are congruent to each other. Things are of course different in the case of spheres. A preliminary list of axioms is reported below.

A14. $CGxy \rightarrow \neg PPxy$

A15. $SPHx \wedge CGxy \rightarrow SPHy$

A16. $(CGxy \wedge IDxyz \wedge EDxyw) \rightarrow CNCzw$

A14 excludes the congruence of two spheres being one internal to the other; A15 states that congruence preserves the property of being a sphere, while A16 allows us to establish the concentricity of two spheres (Figure 12).

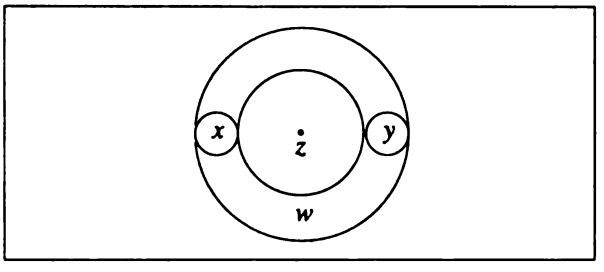


Figure 12. if x and y are congruent then z and w are concentric spheres (A16).

We must now add some *existential axioms*, which are necessary to ensure that enough spheres exist:

A17. $(SEGxy \wedge \neg Oxy \wedge \neg ETxy) \rightarrow \exists z\ (EDxyz)$

A18. $ETxy \rightarrow \exists z\ (IDxyz)$

A19. $SPHx \rightarrow \exists yz\ (IDyzx \wedge ETyz \wedge CGyz)$

A17 guarantees that, given two not connected spheres, a third sphere always exists between them; A18 states that the minimum sphere including two tangent spheres always exists. A19 states that two congruent tangent spheres, internally diametrical to a given one, always exist. Finally, two last axioms guarantee that our space is tridimensional, by stating that four mutually tangent spheres exist, while five mutually tangent spheres do not exist.

A20. $\exists xyzw\ (CGxy \wedge CGxz \wedge CGxw \wedge ETxy \wedge ETxz \wedge ETxw \wedge ETyz \wedge ETyw \wedge ETzw)$

A21. $\forall xyzwv \neg (CGxy \wedge CGxz \wedge CGxw \wedge CGxv \wedge ETxy \wedge ETxz \wedge ETxw \wedge ETxv \wedge ETyz \wedge ETyw \wedge ETyv \wedge ETzw \wedge ETzv \wedge ETwv)$

5.7 USING THE CONGRUENCE PRIMITIVE

We are now in the position to define l- and p-connection (Figure 13), and then the usual notion of connection:

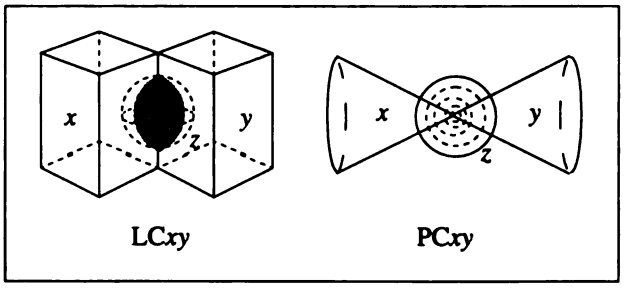


Figure 13. Line– and point-connection.

D29 $LCxy =_{df} \neg SCxy \wedge \exists z(SPHz \wedge Ozx \wedge Ozy \wedge SR(z\text{-}x) \wedge SR(z\text{-}y) \wedge \neg SR(z\text{-}(x+y)))$
(l-connection)

D30. $PCxy =_{df} \neg SCxy \wedge \neg LCxy \wedge \exists z(SPHz \wedge \forall u(CNCuz \rightarrow (Oux \wedge Ouy)))$
(p-connection)

D31. $Cxy =_{df} SCxy \vee LCxy \vee PCxy$ *(connection)*

Congruence allows us to easily define the notion of *convexity* for region (Figure 14):

D32. $CONVx =_{df} \forall uvw(P(u+v)x \wedge CGuv \wedge CGwu \wedge BTWwuv) \rightarrow Pwx$ *(x is convex)*

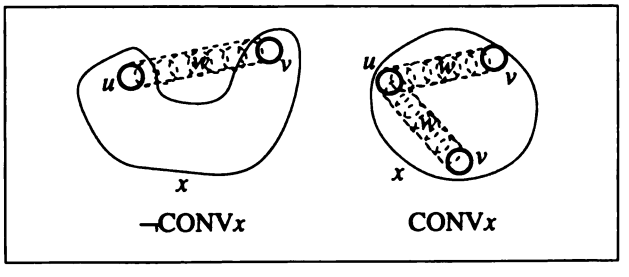


Figure 14. Defining convexity.

Here there are some theorems stating interesting properties of some morphological relations. T18 shows that there

exists only one sphere concentric with y and congruent to x. T19 states that 'ET' relation is preserved by congruence. T20-T23 establish fondamental properties of the 'BTW' relation. T24 shows that spheres are convex regions.

T17. $CNCxy \rightarrow Pxy \vee Pyx$ (D16)

T18. $(SPHx \wedge SPHy) \rightarrow \exists!z(CGzx \wedge CNCzy)$ (A10; T17)

T19. $(SPHx \wedge ETxy \wedge CG(x+y)(x+z)) \rightarrow ETxz$ (A10; A11)

T20. $BTWxyz \leftrightarrow BTWxzy$ (D17)

T21. $\exists yz(BTWxyz)$ (A10; A17; A20; T19)

T22. $BTWxyz \rightarrow (\neg BTWyxz \wedge \neg BTWzyx)$ (A18; A19)

T23. $(BTWyxz \wedge BTWzyu) \rightarrow (BTWyxu \wedge BTWzxu)$ (A10; A16; A17; A18)

T24. $SPHx \rightarrow CONVx$ (A10; A11; D32)

6 DISCUSSION

We have presented a logical theory of space quite rich in axioms and definitions. As stated in the introduction, its main purpose is to characterize the intended meaning of the three primitives used – parthood, simple region, and congruence – in a domain where only three-dimensional regular regions are assumed. However, even if the formal properties of this theory (i.e. its soundness, its completeness and its computational properties) have yet to be studied in detail, the important theorems coming out of the proposed axiomatization let us describe some fundamental features of space.

In this preliminary work, we have concentrated on the task of explaining in detail our ontological assumptions about space by means of formal axioms, exploring at the same time the expressive power, the mutual relationships and the *cognitive relevance* of the primitives adopted.

Let us now add some comments on the comparison between our theory and the RCC theory. We observe first that the latter is a *minimal* theory of connection: this choice may have the advantage of some computational properties, but it is less precise in the meaning of the primitives assumed. Our theory, on the other hand, characterizes more precisely the primitives. In particular, RCC's 'C' primitive can be interpreted as denoting strong-connection, line-connection, point-connection or a combination of these, while we have shown that the above axiomatization prevents the interpretation of our 'SC' as line-connection or point-connection.

This means that the definitions of point-connection, "doughnut" and "quasi-manifold" discussed in (Gotts 1994, Gotts 1994, Cohn 1995) only hold in the *intended* model where two regions are connected if they share at least one point; but if we consider a model were 'C' is interpreted as s-connection then the definitions do not capture the desired meanings. This freedom in the interpretation could be an advantage for the RCC approach, in the sense that the theory is apt to capture a very general notion of connection, which may be useful for various purposes. However, the theory appears to be too weak for a formal characterization of space in its present state.

Full Clarke's theory, as recently shown in (Asher and Vieu 1995), is surely more satisfactory in this respect, but it pays the price of committing to the classical distinction between open and closed regions, which many people consider debatable from the cognitive point of view. Moreover, as discussed in (Varzi 1996b), full Clarke's theory presents some unpleasant mereological properties, since "an open region is always a proper part of its own closure, but there is no mereological difference between the two".

In order to find a satisfatory solution to these problems, our strategy has been to emphasize the role played by the *morphological* properties of space. The definition of sphere (D11) appears to us to be emblematic in this respect. Of course, it requires a characterization of congruence, and this is a complicated task. We admit that the methodology adopted to have an axiomatization of congruence turns out to be complicated and difficult, but unfortunately the alternative would have been "inventing" our axioms for congruence between regions merely by means of introspection. In the literature we are aware of, the only proposal in this sense is the axiomatization of "convex hull" used within the RCC theory, but the same authors admit it cannot be considered as satisfactory yet. The approach we have developed offers a characterization of a *very* powerful primitive and seems to us amenable both for concrete applications and for further mathematical speculations.

Acknowledgements

We are indebted to Tony Cohn, Luis Fariñas del Cerro, Gianni Gerla, Achille Varzi and Laure Vieu for their precious comments on earlier drafts of this paper. This work has been done in a special CNR project "Strumenti Ontologico-Linguistici per la Modellazione Concettuale". The cooperation with Laure Vieu has been done in the CNR/CNRS project "Representing Space and Uncertainty in Intelligent Systems".

References

Asher, N. and Vieu, L. 1995. Toward a Geometry of Common Sense: A Semantics and a Complete Axiomatization of Mereotopology. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI 95)*. Montreal, Morgan Kaufmann Publishers, Inc.: 846-852.

Aurnague, M. and Vieu, L. 1993. A Three Level Approach to the Semantics of Space. In C. Z. Wibbelt (ed.) *The Semantics of Preposition: From Mental*

Processing to Natural Language Processing. Mouton de Gruyter, Berlin: 393-439.

Bennett, B. 1995. Carving Up Space: Existential Axioms for a Formal Theory of Spatial Regions. In *Proceedings of IJCAI-95 workshop on Spatial and Temporal Reasoning*.

Borgo, S., Guarino, N., and Masolo, C. 1996. Towards an Ontological Theory of Physical Objects. In *Proceedings of IMACS-IEEE/SMC Conference "Computational Engineering in Systems Applications" (CESA 96), Symposium on Modelling, Analysis and Simulation*. Lille, France: 535-540.

Casati, R. and Varzi, A. 1995. Basic Issues in Spatial Representation. In *Proceedings of 2nd World Conference on the Fundamentals of Artificial Intelligence*. Paris, Université Paris VI.

Casati, R. and Varzi, A. 1996. The Structure of Spatial Localization. *Philosophical Studies*, **82**: 205-239.

Clarke, B. L. 1981. A Calculus of Individuals Based on "Connection". *Notre Dame Journal of Formal Logic*, **22**: 204-18.

Davis, E. 1993. The kinematics of cutting solid objects. *Annals of Mathematics and Artificial Intelligence*, **9**: 253-305.

Eschenbach, C. and Heydrich, W. 1995. Classical mereology and restricted domains. *International Journal of Human and Computer Studies*, **43**(5/6): 723-740.

Gerla, G. 1994. Pointless Geometries. In F. Buekenhout (ed.) *Handbook of Incidence Geometry*. Elsevier Science Publishers.

Gotts, N. M. 1994. How Far Can We "C"? Defining a "Doughnut" Using Connection Alone. In J. Doyle, E. Sandewall and P. Torasso (eds.), *Principles of Knowledge Representation and Reasoning: KR 94*. Morgan Kaufmann, San Francisco (CA): 246-257.

Gotts, N. M., Gooday, J. M., and Cohn, A. G. 1996. A connection Based Approach to Commonsense Topological Description and Reasoning. *The Monist: An International Journal of General Philosophical Inquiry*, **79**(1).

Grzegorczyk, A. 1960. Axiomatizability of geometry without points. *Synthese*, **12**: 228-235.

Guarino, N. 1995. Formal Ontology, Conceptual Analysis and Knowledge Representation. *International Journal of Human and Computer Studies*, **43**(5/6): 625-640.

Guarino, N., Carrara, M., and Giaretta, P. 1994. Formalizing Ontological Commitment. In *Proceedings of National Conference on Artificial Intelligence (AAAI-94)*. Seattle, Morgan Kaufmann.

Guarino, N. and Giaretta, P. 1995. Ontologies and Knowledge Bases: Towards a Terminological Clarification. In N. Mars (ed.) *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing 1995*. IOS Press, Amsterdam: 25-32.

Hayes, P. 1985a. The Second Naive Physics Manifesto. In J. R. Hobbs and R. C. Moore (eds.), *Formal Theories of the Commonsense World*. Ablex, Norwood, New Jersey: 1-36.

Hayes, P. J. 1985b. Naive Physics I: Ontology For Liquids. In J. R. Hobbs and R. C. Moore (eds.), *Formal Theories of the Commonsense World*. Ablex, Norwood, New Jersey: 71-107.

Lobachevskij, N. I. 1835. New principles of geometry with complete theory of parallels (in Polish). *Polnoe sobranie socinenij*, **2**.

Randell, D. A. and Cohn, A. G. 1992. A Spatial Logic Based on Regions and Connections. In B. Nebel, C. Rich and W. Swartout (eds.), *Principles of Knowledge representation and Reasoning. Proceedings of the Third International Conference*. Morgan Kaufmann, Los Altos: 165-76.

Shanahan, M. 1995. Default reasoning about spatial occupancy. *Artificial Intelligence*, **74**: 147-163.

Simons, P. 1987. *Parts: a Study in Ontology*. Clarendon Press, Oxford.

Smith, B. 1996. Mereotopology: a Theory of Parts and Boundaries. *Data and Knowledge Engineering*, special issue on *Modeling Parts and Wholes* (in press).

Tarski, A. 1956. Foundations of the geometry of solids. In J. Corcoran (ed.) *Logic, semantics, metamathematics*. Oxford University Press, Oxford: 24-30.

Varzi, A. 1994. On the Boundary Between Mereology and Topology. In R. Casati, B. Smith and G. White (eds.), *Philosophy and the Cognitive Science*. Hölder-Pichler-Tempsky, Vienna.

Varzi, A. 1996a. Boundaries, Continuity, and Contact. *Noûs*: (to appear).

Varzi, A. 1996b. Parts, Wholes, and Part-Whole Relations: The Prospects of Mereotopology. *Data and Knowledge Engineering*, special issue on *Modeling Parts and Wholes* (in press).

Whitehead, A. N. 1929. *Process and Reality. An Essay in Cosmology*. Macmillan, New York.

Representing Spatial Vagueness: A Mereological Approach*

Anthony G Cohn & Nicholas Mark Gotts
Division of Artificial Intelligence
School of Computer Studies
University of Leeds, Leeds LS2 9JT, UK
Telephone: +44-113-233-6806
Fax: +44-113-233-5468
Email: {ngotts,agc}@scs.leeds.ac.uk

Abstract

Spatial reasoning is crucial in many AI application domains, such as robotics, qualitative and naive physics, and some types of planning. Qualitative reasoning is often required; and in many of these cases, *uncertainty* or *imprecision* about the spatial extent of particular entities has to be represented and coped with. This paper develops an axiomatisation of a relation of 'crisping' (reducing imprecision or vagueness) between pairs of 'vague' spatial regions: those with indeterminate boundaries. This axiomatisation is then related to the previously developed 'egg-yolk' representation of vagueness, originally designed by (Lehmann and Cohn 1994) for database applications, then applied to expressing spatial vagueness by (Cohn and Gotts 1996).

Topic areas: Representing Space, Reasoning under Uncertainty

1 Introduction: Vague Spatial Regions

> 'Traffic chaos enveloped central Cambridge today, as the KR community gathered from all parts of the industrialised world.'

Where exactly are or were the limits of the traffic chaos, central Cambridge, the industrialised world? They do not exist: some points or areas would be definitely within the regions these entities occupy, some definitely outside, but for many there is no right or wrong answer. How to represent and reason about such entities, and the 'vague' regions they occupy, is the topic of this paper.

Qualitative spatial reasoning is crucial in many AI application domains, e.g. robotics, qualitative and naive physics, and many kinds of planning. Often, there is *uncertainty* or *imprecision* about the spatial extent of particular entities (physical objects, regions defined by some property such as temperature, and/or socially defined regions such as those owned by persons or organizations). We concentrate here on the development of a representational formalism for such vague spatial regions; we consider the development of an adequate representation essential prior to detailed consideration of questions relating to reasoning and applications. We are sceptical about the merits of 'fuzzy' approaches to indeterminacy, believing that their use of real number indices of degrees of membership and truth are both counterintuitive, and logically problematic. We have no space to argue this controversial viewpoint here; see (Elkan 1994) and responses for arguments on both sides.

We need to say at least some of the same sorts of things about vague regions as about 'crisp' ones, with precise boundaries: that one contains another (southern England contains London, even if both are thought of as vague regions), that two overlap (the Sahara desert

* We gratefully acknowledge the support of the EPSRC under grant GR/H/78955 and also the CEC under the Basic Research Action MEDLAR 2, Project 6471. We also acknowledge useful discussions with the rest of the QSR group at Leeds and Eliseo Clementini.

and West Africa), or that two are disjoint (the Sahara and Gobi deserts). In these cases, the two vague regions represent the space occupied by distinct entities, and we are interested in defining a vague area corresponding to the space occupied by either, by both, or by one but not the other. We may also want to say that one vague region is a 'crisper' version of another. For example, we might have an initial (vague) idea of the extent of a mineral deposit, then receive information reducing the imprecision in our knowledge. Here, the vagueness of the vague region is a matter of our ignorance: the entity concerned actually occupies a fairly well-defined region — though perhaps any entity's limits will be imprecise to some degree. In other cases, vagueness appears intrinsic: consider an informal geographical term like 'southern England'. The uncertainty about whether particular places (north of London but south of Birmingham) are included cannot be resolved definitively: it is a matter of interpretational context. A contrasting example is the region occupied by a cloud of gas from an industrial accident. Here we have two sources of intrinsic vagueness: the concentration of the gas is likely to fall off gradually as we move out of the cloud; and its extent will also vary over time, so any temporal vagueness (e.g., if we are asked about the cloud's extent 'around noon') will result in increased spatial vagueness.

In these cases of intrinsic vagueness, there is a degree of arbitrariness about any particular choice of an exact boundary, and often, none is required. But *if* we decide to define a more precise version (either completely precise, or less vague but still imprecise), our choice of version is by no means *wholly* arbitrary: we can distinguish more and less 'reasonable' choices of more precise description.

Distinguishing ignorance-based from intrinsic vagueness is important, but many of the same problems of representation and reasoning arise for both. The distinction between purely spatial entities ('regions'), and what *occupies* such regions (physical objects in the broadest sense, or social phenomena such as areas of legal ownership and political influence), will also be significant. We are concerned mainly with regions here, but need to remember that it is what occupies a region that makes it interesting [1].

Since we present here an axiomatisation which we claim captures important aspects of the intuitive concept of a vague region (and of relations between vague regions, and between vague and crisp ones), we distinguish the intuitive terms 'crisp' and 'vague' from their formally defined equivalents, **Crisp** and **NonCrisp**. When we want to talk about regions which may be either **Crisp** or **NonCrisp**, we will refer to '**OCregions**' (for 'Optionally Crisp regions'). We will use the informal term 'blurring' as the converse of 'crisping'.

Section 2 develops a set of axioms that expresses some of the things we intuitively want to hold about the relations between alternative, more or less vague, estimates of the spatial extent of some extended entity, when this is imprecisely known or defined. We express the relation between two such estimates, of which one is a refinement of the other, as $X \prec Y$, read: 'region X is a crisping of vague region Y'[2]. A parallel is drawn between the axiom-set developed, and the axiom-sets for mereology (theory of part-whole relations) discussed by (Simons 1987). Toward the end of section 2, a start is made on exploring the possible relations between vague regions representing the spatial extent of different entities, revealing considerable complexity. Section 3 introduces the previously developed egg-yolk calclulus which can be used for this latter purpose (Cohn and Gotts 1996), linking it to the approach of section 2 and showing that it can serve to reduce this complexity; it also suggests why the parallels with mereology explored in section 2 exist. Section 4 briefly discusses an expanded version of the egg-yolk formalism. Section 5 discusses future work, including a possible application area.

2 Axioms for the $\prec$ Relation

We assume first order predicate calculus with equality as a logical basis. What sort of properties do we want $\prec$ to have? First, $\prec$ should be asymmetric, irreflexive and transitive:

(A1)$\forall X, Y[X \prec Y \rightarrow \neg Y \prec X]$

(A2)$\forall X, Y, Z[[X \prec Y \wedge Y \prec Z] \rightarrow X \prec Z]$

(A1) ensures $\prec$ is asymmetric and hence irreflexive, (A2) that it is transitive. It will be convenient to add some definitions as part of the basic package:

(D1)$X \preceq Y \equiv_{def} X \prec Y \vee X = Y$

(D2)$X \succ Y \equiv_{def} Y \prec X$

(D3)$\mathsf{MA}(X, Y) \equiv_{def} \exists Z[Z \preceq X \wedge Z \preceq Y]$

(D4)$\mathsf{NMA}(X, Y) \equiv_{def} \neg\mathsf{MA}(X, Y)$

(D5)$\mathsf{Crisp}(X) \equiv_{def} \neg\exists Y[Y \prec X]$

(D6)$\mathsf{NonCrisp}(X) \equiv_{def} \exists Y[Y \prec X]$

(D7)$X \prec\!\!\prec Y \equiv_{def} X \preceq Y \wedge \mathsf{Crisp}(X)$.

$\preceq$ can be read: 'crisper than or equal to'. $\succ$ is the inverse of $\prec$, while **MA** and **NMA** stand for two re-

[1] Here, we allow regions themselves to be vague; it would be possible to assign the vagueness of spatial extent only to physical objects and other 'region-occupiers', or to a function mapping them to sets of Crisp regions instead.

[2] We will use upper-case italic letters for variables ranging over OCregions.

gions being 'mutually approximate' or 'not mutually approximate': i.e, having, or not having, some common region which is $\preceq$ both. Whether Crisp OCregions actually exist can be determined by adding an additional axiom, or left open. $X \prec\!\!\prec Y$ can be read 'X is a completely crisp version of Y', or 'X is a complete crisping of Y'.

These basic axioms and definitions alone allow us to show that some intuitively correct properties of vague regions and crispings hold. For example, we can show that if X and Y are NMA, and Z is a crisping of X, it cannot be MA with Y:

(T1)$\forall X, Y, Z$
$[[\mathsf{NMA}(X,Y) \wedge Z \prec X] \rightarrow \mathsf{NMA}(Z,Y)]]$.

If T1 is not true, then:

(1)$\exists X, Y, Z[\mathsf{NMA}(X,Y) \wedge Z \prec X \wedge \neg\mathsf{NMA}(Z,Y)]$
(2)$\exists X, Y, Z[\mathsf{NMA}(X,Y) \wedge Z \prec X \wedge \mathsf{MA}(Z,Y)]$
(1, D4)
(3)$\exists X, Y, Z[\mathsf{NMA}(X,Y) \wedge Z \prec X \wedge$
$\exists W[W \preceq Z \wedge W \preceq Y]]$ (2, D3)
(4)$\exists X, Y, Z, W[\mathsf{NMA}(X,Y) \wedge Z \prec X \wedge$
$W \preceq Z \wedge W \preceq Y]$(3)
(5)$\exists X, Y, Z, W[\mathsf{NMA}(X,Y) \wedge W \preceq X \wedge W \preceq Y]$
(1, 4, A2)
(6)$\exists X, Y[\mathsf{NMA}(X,Y) \wedge \mathsf{MA}(X,Y)]$ (5, D3)
(7)$\perp$ (6).

At least one further axiom seems necessary, to ensure that, given one crisping of a NonCrisp region, there is an alternative, incompatible one; if a crisping of a region does not mean excluding some alternative possibilities, it is not clear what *could* be meant by one region being a crisping of another. We therefore adopt:

(A3)$\forall X, Y[X \prec Y \rightarrow \exists Z[Z \prec Y \wedge \mathsf{NMA}(X,Z)]]$.

Beyond these three axioms, there are several independent 'dimensions' along which sets of axioms for vague regions can be extended. These can best be explored using a parallel between the relations of crisping and blurring on the one hand, and part/whole relations on the other. If we regard the crisping relation $\prec$ as analogous to a proper part relation, the kinds of distinction made between possible mereological systems by (Simons 1987) are very similar to those that arise in the case of crisping.

2.1 A 'Minimal Extensional Mereology'

Early in his investigation of mereology, Simons (Simons 1987, pp.25-30) constructs a 'minimal extensional mereology', taking as primitive the proper part relation, which he symbolises $\ll$. The logical basis of the system is:

(SA0) Any axiom set sufficient for first-order predicate calculus with identity.

The first two axioms for $\ll$ are (using a different syntax from Simons):

(SA1) $\forall x, y[x \ll y \rightarrow \neg(y \ll x)]$
(SA2) $\forall x, y, z[[(x \ll y) \wedge (y \ll z)] \rightarrow x \ll z]$

These, like (A1) and (A2) above, simply assert that the system's basic relation is a strict partial ordering. Simons goes on to define part (symbolised '$<$'), in a way that parallels our definition of $\preceq$.

Simons' next step is to note that an individual cannot have a *single* proper part, just as we noted that a vague region cannot reasonably have a *single* crisping. He considers various axioms which ensure that if an individual has a proper part, it has at least two. After defining overlapping ('$\circ$') and disjointness (for which we use '$\|$', as Simons' symbol is unavailable), in ways directly corresponding to our definitions of MA and NMA, Simons chooses:

(SA3)$\forall x, y[x \ll y \rightarrow \exists z[z \ll y \wedge z \parallel x]]$.

This axiom he refers to as the *Weak Supplementation Principle* (WSP), and in asserting that any individual with a proper part has another that is disjoint with the first, it corresponds exactly to our axiom A3 above.

The axiom set SA0-3 still permits various models Simons regards as unsatisfactory, in which overlapping individuals do not have a unique product or intersection. One of these is shown as figure 1 (lines join 'parts' (below) to 'wholes' (above)). Such models are ruled out by adding:

(SA6)$\forall x, y[x \circ y \rightarrow \exists z \forall w$
$[w < z \equiv w < x \wedge w < y]]$,

which ensures the existence of such a unique product.

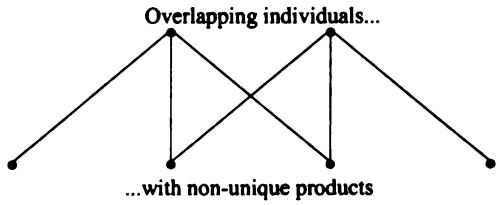


Figure 1: An Inadmissible Model of Part-Whole Relations

The corresponding axiom for $\prec$ ensures the existence of a 'vaguest common crisping' (VCC), such that any other common crisping of the two is also a crisping of the VCC:

(A4)$\forall X, Y[\mathsf{MA}(X,Y) \rightarrow \exists Z \forall W$
$[W \preceq Z \equiv W \preceq X \wedge W \preceq Y]]$.

Thus far, our axiomatisation of $\prec$ corresponds precisely to Simons' minimal extensional mereology, and all the straightforward theorems concerning part, overlap, disjointness and product which Simons lays out

(Simons 1987, pp.38-39) will have their analogues in our system. Once we start considering the 'blurring' process (the inverse of crisping), matters become a little more complex.

2.2 Blurring, and the 'Complete Blur'

Simons notes that his 'minimal extensional mereology' of SA0-3 and SA6 is much weaker than the 'classical' mereological systems of (for example) (Tarski 1956), or (Leonard and Goodman 1940), even for small finite models, because the existence of sums or upper bounds is not guaranteed. (A mereological 'upper bound' (u.b.) of two individuals is an individual of which both are parts; a mereological 'sum' is an u.b. of which any part overlaps one of the two individuals summed.) He therefore considers how u.b.s and sums can be added. Correspondingly, we can consider how to ensure the existence of 'common blurrings' of pairs of **OCregions**. We will not follow Simons the whole way here, as we do not currently try to define *general* blurrings, over arbitrary sets of **OCregions**; only binary ones.

Simons suggests a considerable number of different axioms and combinations of axioms concerning the existence of mereological u.b.s, least u.b.s (an u.b. that is part of any other u.b. — if a sum exists it is always the least u.b., but a least u.b. need not be a sum) and sums, for pairs of individuals. It will be useful to describe his axioms, although we do not write them out:
(SA12) Guarantees the existence of an u.b. for any pair of individuals.
(SA7) Does the same, for pairs of overlapping individuals only.
(SA13) Guarantees the existence of a least u.b. for any pair of individuals.
(SA8) Does the same, for pairs of overlapping individuals only.
(SA14) Guarantees the existence of a sum for any pair of individuals.
(SA9) Does the same, for pairs of overlapping individuals only.
(SA10) Guarantees that a least u.b. exists for any pair with an u.b..
(SA11) Guarantees that a sum exists for any pair of individuals with an u.b..
(SA15) Guarantees that when a least u.b. exists, it is a sum.
(SA16) Guarantees a universal individual exists, which is an u.b. for any pair.

Clearly, a mereology does not need *all* of these: the unconditional existence guarantees (SA12, SA13 and SA14) subsume their conditional counterparts (respectively SA7; SA8 and SA10; SA9 and SA11), any guarantee of the existence of a sum makes the corresponding guarantee of a least u.b. or u.b. unnecessary, and if a least u.b. exists then so does an u.b.. SA14 in fact subsumes all of SA7-13 and SA15. SA16, guaranteeing the existence of a universal region, is independent of all the others apart from SA7 and SA12, which it subsumes, but which are weaker. Adding it to SA0-3, SA6 and SA14 creates a Boolean algebra minus a zero element. This is the strongest mereological system Simons proposes without the use of *general* least u.b.s or sums.

The analogue of guaranteeing the existence of an u.b. is guaranteeing that of a 'common blurring' (**CB**) for a pair of **OCregions**: an **OCregion** to which both members of the pair are $\preceq$. A least u.b. would correspond to a 'crispest common blurring' (**CCB**) of a pair of **OCregions**, and a sum to an **OCregion** which is a crispest common blurring of which *any* crisping is **MA** to one of the pair (we could call it a 'blur sum' or **BS**). The analogue of the universal region of SA16 could be called the 'complete blur': a **NonCrisp** region which is a blurring of any other **OCregion**.

Which of these analogues are plausible? In particular, would we want to adopt analogues of SA14 and SA16? If not, what weaker substitutes should be adopted? Figure 2a shows why we do *not* want to adopt an analogue of SA14. In the left part of figure 2a, the inner and outer pairs of circles represent two **NonCrisp** regions: we might take the inner of each pair to represent an area definitely within the **NonCrisp** region, the outer an area definitely containing it. Imagine that these are, for example, two versions of the area enveloped by a flood received from different sources, each with some imprecision. What might their 'crispest common blurring' (**CCB**) look like? The right side of figure 2a shows one possibility: we retain the inner circle from the smaller of the pair, and the outer from the larger, and take these as representing inner and outer limits on a vaguer **NonCrisp** region which can be regarded as a blurring of both. However, the dashed circles suggest another possible crisping of this **CCB**, having *no* common crisping with either of the original pair. If this is admitted, then the **CCB** *cannot* be the pair's **BS** ('blur sum'): any crisping of the **BS** of two **OCregions** should be **MA** with one of the pair, just as any part of a mereological sum of a pair of individuals overlaps one of them.

If a **CCB** must exist for any pair of **OCregions**, but a **BS** may not, we need an analogue of SA13. We therefore adopt:

(A5)$\forall X, Y[\exists Z[X \preceq Z \wedge Y \preceq Z \wedge$

Figure 2: 'Crispest Common Blurrings'

$\forall W[X \preceq W \wedge Y \preceq W \rightarrow Z \preceq W]]]$.

Depending on our interpretation of OCregions, we might even want to deny that a CCB must exist for any pair. Consider figure 2b: the lighter and darker circles represent the inner limits of two NonCrisp regions with a common outer limit (the containing oval). If we allow inner and/or outer limiting areas of a NonCrisp region to be discontinuous (multi-piece) — as we currently wish to do — then the CCB of these two NonCrisp regions can be arrived at by adding the two inner limiting regions to get the CCB's inner limiting region, and letting their common outer limit be the CCB's outer limit. However, if we wanted to disallow discontinuous inner limiting regions, the inner limiting region of any u.b. of the two NonCrisp regions must include their inner limiting regions *and* a 'bridge' between them, such as those outlined by the upper, dashed pair of lines, or the lower, dotted ones. Clearly in this case there would be no unique such bridge and hence no CCB.

The argument above concerning figure 2a suggests that two crispings of an OCregion should have a CCB *if and only if* they are MA. This is analogous to a mereological system in which a pair of individuals always have a least upper bound, but have a sum if and only if they overlap. In addition to (A5), therefore, we adopt:

(A6)$\forall X, Y[\mathsf{MA}(X,Y) \equiv \exists Z[[X \preceq Z \wedge Y \preceq Z \wedge$
$\forall W[X \preceq W \wedge Y \preceq W \equiv Z \preceq W]]]]$.

Note that (A6) does not correspond precisely to any of Simons' axioms: it is stronger than the analogue of SA9, which would guarantee the existence of a BS for any pair of MA OCregions, but would not rule out the existence of a BS for NMA pairs.

Whether we adopt an analogue of SA16 depends on the view we take of OCregions. One interpretation of a 'complete blur', a vague region which is a blurring of any other, is that *no* area is either definitely in such a region, or definitely out of it: for example, we might want to represent (our current state of knowledge concerning) the parts of the universe which have at any time been inhabited by extraterrestrial beings. This is our current preference, and we thus adopt:

(A7)$\exists X \forall Y[Y \preceq X]$.

If we only wanted to talk about NonCrisp regions with some limits on their blurring, we could adopt the negation of this:

(A7a)$\neg\exists X \forall Y[Y \preceq X]$[3].

To specify necessary and sufficient conditions for the identity of two OCregions, we need an axiom linking $\prec$ and $=$. We could choose:

(A8a)$X = Y \equiv \forall Z[X \prec Z \equiv Y \prec Z]$.

(X and Y are equal iff (X is a crisping of any Z iff Y is a crisping of that Z).) If we wanted to *define* equality in terms of $\prec$, rather than assuming it, we could make this a definition. Note that this axiom implies that there is at most one distinct OCregion with no blurrings (the complete blur, if it exists). If we axiomatised equality for OCregions instead as:

(A8b)$X = Y \equiv \forall Z[Z \prec X \equiv Z \prec Y]$,

this would require that there would be at most one OCregion with no crispings, that is, at most one Crisp region. In fact, there is an alternative we prefer to either of these:

(A8)$X = Y \equiv \forall Z[Z \prec\!\!\prec X \equiv Z \prec\!\!\prec Y]$

(any *complete* crisping of either OCregion is also a complete crisping of the other).

2.3 Do Crisp Regions Exist?

Do Crisp OCregions exist? We have four alternative possibilities, according to whether A9, A9a, A9b, or none of the three is included in the set of axioms:

(A9)$\forall X[\exists Y[Y \preceq X \wedge \mathsf{Crisp}(Y)]]$.

(A9a)$\forall X[\exists Y[Y \prec X]]$

(A9b)$\exists X[\mathsf{Crisp}(X)] \wedge$
$\exists X \forall Y[Y \preceq X \rightarrow \exists Z[Z \prec Y]]$

The first of these asserts that all regions are Crisp, or have a complete crisping; the second that there are no Crisp regions. The third asserts that some regions are 'crispable', while others are not.

The existence or non-existence of Crisp regions is analogous to that of mereological 'atoms': individuals without proper parts. Here, bearing in mind the distinction made between spatial regions and the entities that occupy them, we will add A9 to our set of axioms: all NonCrisp OCregions are crispable. We further assume that for every pair of OCregions such that one is a crisping of the other, there is a third which lies between them:

(A10)$\forall X, Y[X \prec Y \rightarrow \exists Z[X \prec Z \wedge Z \prec Y]]$.

This is a kind of 'denseness' axiom. Its mereological counterpart would assert that for any two regions of which one is a proper part of the other, there is a third which is a proper superpart of the first and a proper part of the second.

[3] Axioms belonging to our current axiom-set are numbered from (A1) to (A12). Possible alternatives are given an additional lower-case letter, e.g. (A7a).

In contrast to *regions*, we will allow that some physical and social entities are 'uncrispable'. In a fuller theory, linking spatial regions and entities with additional, non-spatial properties, we would need a function mapping things with spatial extent onto the regions they occupy. Some types of entities might be restricted to mapping onto **NonCrisp OCregions**; others, to **Crisp OCregions**. Furthermore, given a temporal theory, we could formally express the fact that for some kinds of physical entities, the regions would become crisper (or at least not less crisp) as our knowledge of their spatial extent grew, whilst for others, their crispness might not necessarily increase with time (e.g. the spatial extent of a river which varies over time).

Asserting that all **NonCrisp OCregions** are crispable gives us a way to deal with spatial relations between the entities associated with vague regions, by thinking in terms of the relations between members of their sets of complete crispings. We explore this approach here for the 'proper part' relation. As we noted earlier, we want to say that 'London' is a proper part of 'southern England', even though both are vague. We can now say that *any* complete crisping of the **OCregion** occupied by London is a proper part of *any* complete crisping of the **OCregion** occupied by southern England. We formally define the relation between the **NonCrisp OCregions** for London and southern England as follows:

$$(\text{D8}) X \ll_{\text{OC}} Y \equiv_{def} \forall Z, W[[Z \prec\!\!\!\prec X \wedge W \prec\!\!\!\prec Y] \rightarrow Z \ll W].$$

Relations such as $\gg_{\text{OC}}$ ($\gg$ is the inverse of $\ll$), $\circ_{\text{OC}}$, $\|_{\text{OC}}$ can be defined analogously, and will have the same properties as their **Crisp** region counterparts: for example, $\ll_{\text{OC}}$ will be asymmetric and transitive.

We can also define weaker versions of the proper part relation between **OCregions** although these are more complicated, e.g.:

$$(\text{D9}) X \ll_{\text{OC}_a} Y \equiv_{def} \forall Z[Z \prec\!\!\!\prec X \rightarrow \exists W[W \prec\!\!\!\prec Y \wedge Z \ll W]] \wedge \forall Z[Z \prec\!\!\!\prec Y \rightarrow \exists W[W \prec\!\!\!\prec X \wedge W \ll Z]].$$

This relation might hold where the stronger does not, for example between the **OCregions** for 'the wettest parts of Britain' and 'the wetter parts of Britain'. However we crisp the second, there will be a complete crisping of the first that is a proper part of it, and however we crisp the first, there will be a proper superpart complete crisping of the second, but *every* complete crisping of the first may not be a proper part (or even a part) of *every* complete crisping of the second. The relation $\ll_{\text{OC}_a}$ and its inverse are still asymmetric and transitive. In fact, there are at least *nine* possible weakened versions of $\ll_{\text{OC}}$, of which *eight* retain this property. Figure 3 shows the relationship between $\ll_{\text{OC}}$ and the nine weaker possibilities: 3a illustrates the logical relationships between six assertions about pairs of *sets* of individuals, members of which may be related by $\ll$. '$\forall A \forall B[A \ll B]$' stands for the assertion that every member of one set (the As) is a proper part of every member of the other (the Bs); '$\exists A \forall B[A \ll B]$' means that there is some member of the As that is a proper part of every member of the Bs, and so on. Straight lines connect stronger assertions (above) to weaker (below). Figure 3b shows all the ten logically distinct conjunctions that can be produced from these six assertions. Each of the ten corresponds to a possible version of $\ll_{\text{OC}}$, with the Xs and Ys being the sets of complete crispings of two **OCregions**: only the one at the bottom fails to retain transitivity.

3 The Egg-Yolk Theory

Given all these possibilities for generalizing mereological relationships from crisp to vague regions, an alternative approach is worth considering. (Lehmann and Cohn 1994) suggest an approach to spatial vagueness that involves using two (or more) concentric subregions, indicating degrees of 'membership' in a vague region. (In the simplest, two-subregion case, the inner is called the 'yolk', the outer the 'white', and the inner and outer subregions together the 'egg'.) Lehmann and Cohn first suggested the egg-yolk approach in the context of the problem of integrating heterogeneous databases, where the notions of 'regions' and 'spatial relations' are used metaphorically to represent sets of domain entities, and their relations. It was developed as an approach to expressing spatial vagueness itself in (Cohn and Gotts 1996). Here, we employ it as a means to understanding the alternative approach we develop.

The egg-yolk formalism as developed in (Lehmann and Cohn 1994) allows for just 5 'base relations' (**DR**, **PO**, **PP**, **PPi** and **EQ**) between any egg-egg or yolk-yolk pair, or any egg and the yolk belonging to another egg. (A yolk is always a **PP** of the corresponding egg.) **EQ**, **PP**, **PPi** and **DR** correspond to Simons' $=$, $\ll$, $\gg$, and $\|$, except that they are defined in terms of a primitive $\mathsf{C}(x, y)$ (connection) (Randell, Cui and Cohn 1992), with which we need not be concerned. **PO** (partial overlap) is simply the relation that holds when none of the other four do: the five base relations (henceforth 'RCC-5' for historical reasons) form a jointly exhaustive and pairwise disjoint (JEPD) set: exactly one holds between any pair of (crisp) regions.

RCC theory asserts the existence of a universal region, **U**s (of which every other region is a **PP**), and provides quasi-Boolean functions (quasi-Boolean because

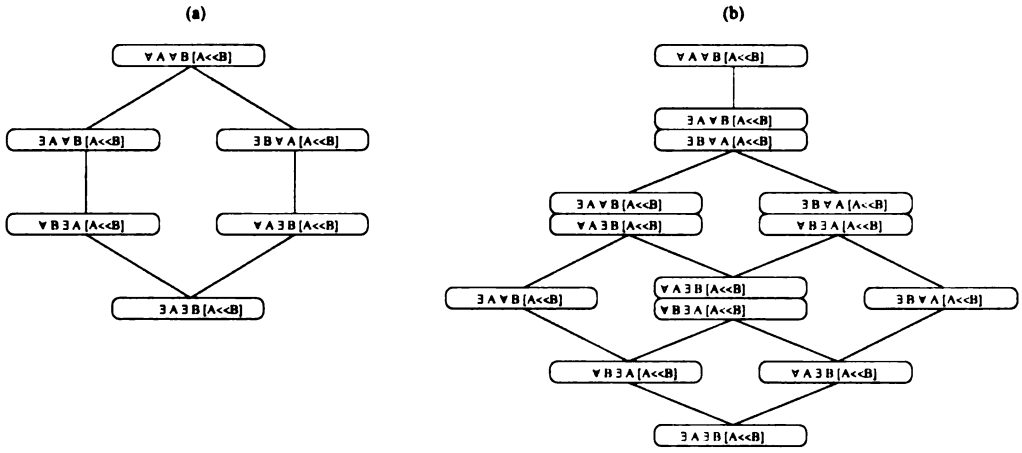


Figure 3: Possible assertions about $\ll$ between two sets of individuals

no 'empty' or 'zero' region exists) on pairs of regions: compl(x) (the complement of a region in Us), and the region-sum, region-intersection, and region-difference of two regions (sum(x, y), prod(x, y) and diff(x, y)). All these functions other than sum may return a NULL object instead of a region (e.g. prod will do so if the two argument regions are DR). The distinction between regions and NULL is dealt with using the sorted logic LLAMA (Cohn 1987).

The RCC-5 set produces 46 possible relations between a pair of egg-yolk pairs (see figure 4). In (Cohn and Gotts 1996) we argued that these 46 could be identified with the possible relations between complete crispings of two vague regions. Here, we take the OCregions of section 2 as our initial representation of vague regions, and show how the egg-yolk representation can be used to provide additional constraints for OCregion theory.

At first glance, there is one apparent problem with the egg-yolk approach: the most obvious interpretation is that it replaces the precise dichotomous division of space into 'in the region' and 'outside the region' of the basic RCC theory by an equally precise trichotomous division into 'yolk', 'white' and 'outside' — and this appears contrary to a key intuition about vagueness: that not only is there a 'doubtful' zone around the edges of a vague region, but that zone itself has no precise boundaries. So we want a way of using the egg-yolk formalism that is consistent with this.

We link the OCregions of section 2 (and the corresponding theory), with ordered pairs of RCC-5 regions, the first of the pair being a part, but not necessarily a proper part, of the second. If it *is* a PP, then the pair is an egg-yolk pair in the sense of 2 (Lehmann and Cohn 1994), and the OCregion is NonCrisp. If not, the OCregion is Crisp. We now link the $\prec$ predicate of OCregion theory with the egg-yolk approach. We define a function ey to map an OCregion to an egg-yolk pair, and two functions eggof and yolkof, to map such egg-yolk pairs to the RCC-5 region comprising its egg and yolk respectively. We will normally write ey(X) as $\hat{X}$ for notational convenience. We have the following axiom for egg-yolk pairs:

(A11) $\forall X$P(yolkof($\hat{X}$), eggof($\hat{X}$))

We then assert the following additional axiom concerning $\prec$:

(A12)$\forall X, Y[X \prec Y \rightarrow$
[[PP(eggof($\hat{X}$), eggof($\hat{Y}$))$\wedge$
P(yolkof($\hat{Y}$), yolkof($\hat{X}$))]$\vee$
[P(eggof($\hat{X}$), eggof($\hat{Y}$))$\wedge$
PP(yolkof($\hat{Y}$), yolkof($\hat{X}$))]]].

This axiom links $\prec$ to the predefined RCC-5 relations by an *implication*, not an *equivalence*: we do *not* specify that if the specified RCC relations hold between eggof($\hat{X}$), yolkof($\hat{X}$), eggof($\hat{Y}$) and yolkof($\hat{Y}$), the CR relation holds between X and Y, but these relations *must* hold for the $\prec$ relation to do so. We leave undefined what additional conditions, if any, must be met. This gives us the kind of indefiniteness in the *extent* of vagueness, or 'higher-order vagueness', that intuition demands. Consider the vague region 'beside my desk'. This can be regarded in OCregion theory as a NonCrisp region. There are some precisely defined regions, such as a cube $10cm$ on a side, $5cm$ from the right-hand end of my desk, and $50cm$ from the floor, that are undoubtedly contained within any reasonable complete crisping of this NonCrisp region. Others, such as a cube $50m$ on a side centred at the front, top right-hand corner of the desk, contain any such reasonable crisping. These two could correspond to the 'yolk' and 'egg' of an egg-yolk pair constituting the NonCrisp region 'beside my desk', forming a very conservative inner and

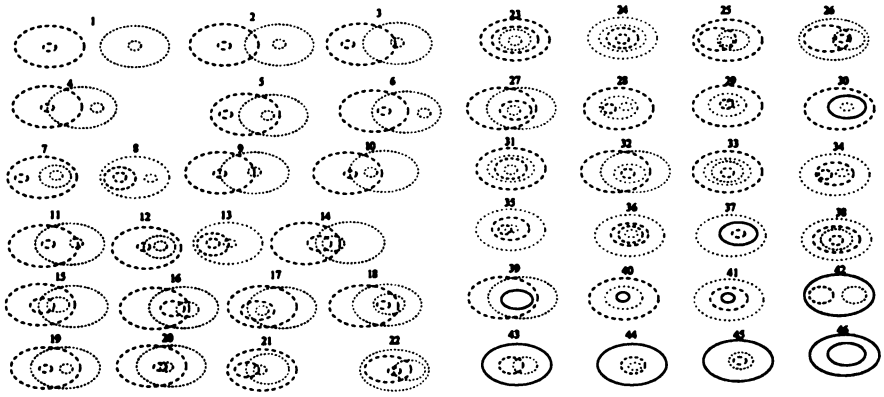


Figure 4: The 46 possible relationships between two egg-yolk pairs

outer boundary on its possible range of indefiniteness. However, some OCregions (Crisp and NonCrisp) lying between this pair would *not* make a reasonable crisping of this region: consider a volume including the 'yolk' of the pair, plus a layer one centimetre deep at the very top of the white. This meets all the conditions for a crisping of the specified OCregion, but is an absurd interpretation of 'beside my desk'. In general, we need not precisely specify the limits of acceptability. For specific applications, we could add further conditions on acceptable crispings (such as preserving particular topological features or relative proportions in different dimensions), and perhaps assert that (for that application) these conditions are sufficient.

Configuration 1 in figure 4, given the interpretation of OCregion region theory in terms of egg-yolk pairs of RCC-5 regions outlined here, clearly shows a pair of NonCrisp regions such that any pair of complete crispings of the two must be DR. Taking the left-hand egg-yolk pair as representing NonCrisp region X, and the righthand one NonCrisp region Y:

$$\forall V, W[[V \prec\!\!\prec X \wedge W \prec\!\!\prec Y] \rightarrow \mathsf{DR}(\mathsf{eggof}(\hat{V}), \mathsf{eggof}(\hat{W}))].$$

Similarly, configuration 2 represents a pair of NonCrisp regions such that, for any complete crisping of either, we can choose a complete crisping of the other that is DR from it, and there are also some complete crisping pairs of the two that are PO. (Cohn and Gotts 1996) shows how each of the 46 configurations can be distinguished in terms of the possible results of replacing one or both of the egg-yolk pairs with a single region-boundary lying within the white of the egg — a complete crisping of the vague region represented by the egg-yolk. To give an example, the most difficult pair of configurations to distinguish in this way are 39 and 43. Completely crisping both egg-yolk pairs may give any of the RCC-5 relations between the complete crispings, other than DR. If either the dashed or the dotted region is completely crisped first, there is no single RCC-5 relation that can then *always* be acheieved by crisping the other. However, the two configurations can be distingushed by considering the *sets* of possibilities that remain after completely crisping (say) the dashed region. In the case of configuration 39, it will always then be possible to completely crisp the dotted region so as to produce one member of *each* of the following sets: {EQ,PO}, {EQ,PPi}, {PP,PO}, {PP,PPi}. The corresponding sets for configuration 43 are {EQ,PO}, {EQ,PP}, {PPi,PO}, {PP,PPi}.

The egg-yolk way of interpreting OCregion theory explains why we found so many parallels with Simons' mereology. Under the egg-yolk interpretation, an OCregion amounts to a three-way division of Us into yolk, white, and non-egg. If we consider a particular NonCrisp region, the complete set of its possible crispings can be regarded as a mereological system, in which the complete crisping constitute the atoms. Crisping expands the yolk and/or non-egg at the expense of the white; in the limit the 'white' becomes a lower-dimensional boundary between yolk and non-white — in other words, the region becomes Crisp. One OCregion in this set being a crisping of another is like one individual being a proper part of another because the white of the first is a proper part of the white of the second. We have a plausible candidate for the VCC of two MA OCregions: the VCC's yolk could be the sum of the yolks of its two blurrings, its egg the prod of the two blurrings' eggs (which, if the two are MA, must exist as a region). Similarly, the yolk of the CCB of *any* two OCregions might be defined as the prod of their yolks; its egg as the sum of their eggs.

The implications of these identifications remain to be explored. However, the egg-yolk model of the OCregion axioms does appear to provide a straightforward way to define OCregion extensions of the compl, sum, prod and diff functions defined within RCC. Using forms such as '$\langle \mathsf{eggof}(X), \mathsf{yolkof}(X) \rangle$'to represent the egg-yolk pairs of RCC-5 regions corresponding to NonCrisp OCregions, we extend the definitions of compl, sum, prod and diff as follows:

(D10)compl($\hat{X}$) $=_{def}$
⟨compl(yolkof($\hat{X}$)), compl(eggof($\hat{X}$))⟩
(D11)sum($\hat{X}, \hat{Y}$) $=_{def}$
⟨sum(eggof($\hat{X}$), eggof($\hat{Y}$)),
sum(yolkof($\hat{X}$), yolkof($\hat{Y}$))⟩
(D12)prod($\hat{X}, \hat{Y}$) $=_{def}$
⟨prod(eggof($\hat{X}$), eggof($\hat{Y}$)),
prod(yolkof($\hat{X}$), yolkof($\hat{Y}$))⟩
(D13)diff($\hat{X}, \hat{Y}$)) $=_{def}$ prod($\hat{X}$, compl($\hat{Y}$)).

In the cases of (D12) and (D13), one or both of the components of the 'output' egg-yolk pair may be NULL (as indeed is also the case with (D10) when taking the complement of the 'complete blur' guaranteed by (A7)). These cases require further investigation to ensure their correct formal treatment.

Figure 5 shows the 46 possibilities assuming that RCC-5 calculus is used to relate eggs and yolks. We will briefly investigate here how egg-yolk theory can be used to explore the relations between vague regions expressed in figure 3. Table 1 shows the various sets of egg-yolk configurations which satisfy the upper five quantificational schema of figure 3a, but considering three of the other four mereological relations along with the proper part relation (we omit PPi as it is simply the dual of PP). Only PO differentiates all five quantificational schema. DR, PP (and PPi) collapse the distinction between ∃X∀Y and ∀Y∃X (and dually, between ∃Y∀X and ∀X∃Y). EQ only distinguishes two cases. Thus, in the egg-yolk interpretation there are not quite as many possible relations as figure 3 might suggest. Moreover, egg-yolk theory gives us a way to reason with vague regions using the existing mechanism of the RCC calculus.

4 An RCC-8 Analysis

The egg-yolk theory described in the previous section relied upon the RCC-5 calculus described in (Lehmann and Cohn 1994). However, the topological part of RCC theory is normally presented as having eight JEPD relations, see figure 5, known as RCC-8 (Randell et al. 1992). We have recently completed an analysis of RCC-8 in the style of (Cohn 1995). There is not space here to present the full set of data and the ensuing analysis, but we can survey these results.

Whereas there were 46 configurations in the RCC-5 case, there are 601 configurations if each yolk may either be an NTPP or a TPPof its egg. If we form these configurations into equivalence classes by not distinguishing whether each yolk is an NTPP or a TPPof its egg, then there are 252 configurations. The following data all refers to this collapsed set of configurations

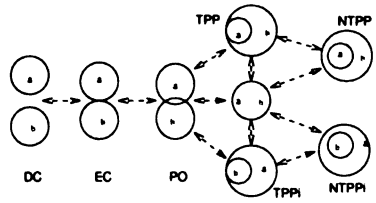


Figure 5: The set of relations in the RCC-8 calculus and their *conceptual neighbourhood* graph (i.e. the continuous transitions between these relations).

(obviously, each collapsed configuration may represent between 1 and 4 of the original 601 configurations). As in the RCC-5 analysis, we can cluster these configurations into maximal groups such that each configuration in each cluster can be crisped into each other configuration in the cluster. It turns out that there are 9 clusters of size 1, 12 clusters of size 4, 4 clusters of size 5, 6 clusters of size 6, 5 clusters of size 9, 1 cluster of size 16, 2 clusters of size 21 and 1 cluster of size 36, making a total of 40 clusters altogether. Each of these clusters has a unique set of complete crispings (i.e. the union of the set of complete crispings of each of the configurations in the cluster). These are displayed below (the number in parentheses after each set is the number of configurations in the cluster).

{DC}(1)
{DC,EC}(4)
{EC}(1)
{DC,EC,=,NTPP,NTPPi,PO,TPP,TPPi}(6)
{DC,EC,=,NTPP,PO,TPP,TPPi}(5)
{EC,=,NTPP,NTPPi,PO,TPP,TPPi}(6)
{EC,=,NTPP,PO,TPP,TPPi}(5)
{=,NTPP,NTPPi,PO,TPP,TPPi}(36)
{=,NTPP,PO,TPP,TPPi}(21)
{DC,EC,=,NTPPi,PO,TPP,TPPi}(5)
{DC,EC,=,PO,TPP,TPPi}(4)
{EC,=,NTPPi,PO,TPP,TPPi}(5)
{EC,=,PO,TPP,TPPi}(4)
{=,PO,TPP,TPPi}(16)
{=,NTPPi,PO,TPP,TPPi}(21)
{DC,EC,NTPP,PO,TPP}(9)
{EC,NTPP,PO,TPP}(6)
{=,NTPP,TPP}(4)
{NTPP}(1)
{NTPP,PO,TPP}(9)
{NTPP,TPP}(4)
{DC,EC,NTPPi,PO,TPPi}(9)
{EC,NTPPi,PO,TPPi}(6)
{=,NTPPi,TPPi}(4)
{NTPPi}(1)
{NTPPi,PO,TPPi}(9)
{NTPPi,TPPi}(4)

	∀X∀Y	∃X∀Y	∀Y∃X	∃Y∀X	∀X∃Y
PP	24	18,24,26,32, 33,37,38,45	18,24,26,32, 33,37,38,45	8,13,22,24,26,34, 35,36,37,38,41	8,13,22,24,26,34, 35,36,37,38,41
EQ	∅	∅	33,40,45,46	∅	36,41,44,46
DR	1	1,2,3,5,7	1,2,3,5,7	1,2,4,6,8	1,2,4,6,8
PO	14	3,5,7,9,10,11,12,14, 15,16,18,19,20,21,28,29,33	3,5,7,9,10,11,12,14,15, 16,18,19,20,21,28,29,32,33	4,6,8,9,10,11,13,14,15, 16,17,19,20,22,34,35,36	4,6,8,9,10,11,13,14,15,16, 17,19,20,22,27,34,35,36

Table 1: The egg-yolk configurations which satisfy various relations as defined by various quantificational schema.

{DC,EC,PO}(9)
{DC,EC,PO,TPPi}(6)
{EC,PO}(4)
{EC,PO,TPPi}(4)
{PO}(1)
{PO,TPPi}(4)
{DC,EC,PO,TPP}(6)
{EC,PO,TPP}(4)
{PO,TPP}(4)
{=,TPP}(1)
{TPP}(1)
{=,TPPi}(1)
{TPPi}(1)

Again, as in the RCC-5 case, each set of complete crispings forms a connected subgraph of the conceptual neighbourhood diagram, however not all connected subgraphs (100 possibilities) are represented by a cluster (just as in the RCC-5 case). Moreover, the links that can be drawn from one cluster to another, which represent the fact that one cluster contains a configuration which can be crisped to a configuration in the other cluster, again coincide with the links that can be drawn from one cluster to another representing superset relations between the sets of complete crispings for each cluster. As in the RCC-5 case, there is just a single cluster (whose complete crispings are the complete set of RCC relations) which has no inward links. In the RCC-8 case there are 7 clusters which have no outward links (each having a singleton complete clustering set, omitting equality as in the RCC-5 case) leaving 32 with both inward and outward links.

We intend to prepare a full report on the RCC-8 analysis in due course. Here we will conclude this brief presentation by noting that the problem noted in (Cohn and Gotts 1996) of representing the fact that two vague regions are externally connected (e.g. a hill and a valley) is addressed by the RCC-8 version of the egg/yolk calculus: there is a unique configuration whose only complete crisping is EC: each yolk is a TPP of its egg, and all the other relationships between eggs and yolks are EC.

5 Discussion: From Representation to Reasoning and Application

The paper has shown how two approaches to the representation of vague spatial regions, originally developed in competition with each other (although partly within the same research team), can be brought together, shedding new light on both and on the representational problems they were designed to solve. The next stage of this research would be to investigate the potential of the two approaches, individually and in combination, to support *reasoning* about vague regions in particular contexts. This requires investigation of the additional conditions that would be required of a reasonable crisping of a NonCrisp region (on top of those imposed by the egg-yolk interpretation) in particular domains. Relevant work on a closely related representation of vague regions is reported in (Clementini and Di Felice 1996).

The mereologically-based approach outlined in section 2 has concentrated on capturing aspects of the relations between different, more or less vague versions of the same region. Here, we have shown that these different versions of a region can be regarded as forming a mereological structure, but most of the axioms adopted simply assert the existence (or, in the case of (A6), nonexistence) of one vague region given the existence of others: more work is required on developing axioms which constrain the relations that can exist between different versions of a region in useful ways.

The other kind of relation we want a vague region formalism to support is that between different spatially extended entities, each occupying a vague region. For example, we want to be able to deduce that if London is a part of southern England, and southern England is disjoint from central Europe, then London is disjoint from central Europe — even if we regard all three as having indeterminate boundaries. For this kind of reasoning, we could make use of relations such as $\ll_{OC}$, defined toward the end of section 2; as noted there, $\ll_{OC}$ (and its inverse) are transitive. However, the link from the mereologically-based formalism to the egg-yolk approach made in section 3 makes available the resources developed within the larger RCC theory of spatial reasoning of which the latter forms a part

(Randell et al. 1992). These include 'transitivity' or 'composition' tables of relations (Cui, Cohn and Randell 1993), which specify the set of possible relations between regions x and z, given those between the pairs x and y, and y and z.

It is no accident that our motivating examples in this paper have been geographical: dealing with geographical regions with indeterminate boundaries is an important issue in current work on GIS (geographical information systems), as the recent meeting specifically on that issue (Cohn and Gotts 1996) indicates. Digital mapping, satellite photography, and GIS have made greatly increased amounts of information available to geographers in recent years; the difficulty they face is to make effective use of this information. Appropriately organised summaries of what is known about spatial relationships of interest — and of what remains uncertain or ill-defined — are of central importance in making available information useful in hypothesis-building and testing, and it is here that we see work on the topology of regions with indeterminate boundaries as potentially of great value. Consider, for example, the questions surrounding global climate change, forest destruction and desertification, and similar actual or possible large-scale environmental problems. Information about the spatial properties and relations of many different kinds of entities and variables, from many different sources of varying reliability, is potentially relevant to unravelling the complex causal interactions involved.

To make the example more specific, what are the leading causes of the degradation and destruction of tropical forests? Logging, mining, road-building, cattle-ranching and cash crops for export, fuelwood gathering, increasing population, the movement of displaced people, government-promoted migration, patterns of land-ownership and the legal framework regulating it, measures purportedly aimed at conservation such as the declaration of national parks — all these and more have been blamed in one region or another (Hecht and Cockburn 1989, Brown and Pearce 1994). Complex interactions between factors differing from region to region are involved. To design and implement effective responses, a qualitative understanding of such interactions is necessary (though certainly not sufficient, as powerful economic and political interest groups are involved). For such an understanding to be developed, it is necessary to know how the spatial distributions of the putative causal factors relate to each other, and to the pattern of destruction in various parts of the world. Some of these factors have well-defined and readily determined spatial boundaries: land ownership and the laws regulating it, for example; but the majority do not.

In an initial search for plausible hypotheses, we would want to ensure that we do not miss potentially important possibilities, and therefore to make generous estimates of the maximum spatial extent of various possible causative factors. A large number of factors might then be shown as possibly overlapping the areas of rapid destruction. However, we would also wish to avoid erroneously concluding that a particular factor is essential, and would therefore wish to have safe minimum estimates of these factors' spatial extent as well. Given similar maximum and minimum estimates of the area of rapid forest destruction, we would then have a (probably large) set of qualitative hypothesis about the causally relevant factors. If we assume that all the important factors lie among those we are considering, then any Boolean combination of factors whose spatial extent could coincide with the areas of rapid destruction (under some crisping of the vague regions concerned) is a possible qualitative hypothesis.

It would then be useful to know the effect on this set of hypotheses of tightening the limits on the spatial extent of the areas of rapid destruction, and/or one or more of the putative causal factors, in order to help us decide what additional information would be most useful in distinguishing between them. To simplify matters, suppose we had just two possible factors in mind, and the areas of rapid destruction lay within the areas covered by factor A whatever estimates we used, but within the areas covered by factor B if and only if we used a narrow definition of the areas of rapid destruction, and a generous estimate of the areas affected by that factor. Additional information about the spatial extent of factor B's effect is more likely to be immediately useful in narrowing the range of hypotheses than information about that of factor A. To reach this conclusion, we must ourselves be employing some intuitive 'logic of vague regions'. Assigning essentially arbitrary numerical measures of 'degree of membership' to particular points, as in a 'fuzzy' approach, appears to us unlikely to be helpful. Clearly, however, the non-numerical approach we have begun to develop here needs considerably more work before its usefulness can be assessed.

References

Brown, K. and Pearce, D. (eds): 1994, *The Causes of Tropical Deforestation*, UCL Press.

Clementini, E. and Di Felice, P.: 1996, Approximate topological relations, *Technical Report R.96-07*,

Università di L'Aquila, Dipartimento di Ingegneria Elettrica.

Cohn, A. G.: 1987, A more expressive formulation of many sorted logic, *Journal of Automated Reasoning* **3**, 113–200.

Cohn, A. G.: 1995, A hierarchcial representation of qualitative shape based on connection and convexity, *in* A. Frank (ed.), *Proc COSIT95*, LNCS, Springer Verlag, pp. 311–326.

Cohn, A. G. and Gotts, N. M.: 1996, The 'egg-yolk' representation of regions with indeterminate boundaries, *in* P. Burrough and A. M. Frank (eds), *Proceedings, GISDATA Specialist Meeting on Geographical Objects with Undetermined Boundaries*, Francis Taylor, pp. 171–187.

Cui, Z., Cohn, A. G. and Randell, D. A.: 1993, Qualitative and topological relationships in spatial databases, *in* D. Abel and B. C. Ooi (eds), *Advances in Spatial Databases*, Vol. 692 of *Lecture Notes in Computer Science*, Springer Verlag, Berlin, pp. 293–315.

Elkan, C.: 1994, The paradoxical success of fuzzy logic, *IEEE Expert* **9**(4), 3–8. Followed by responses and a reply.

Hecht, S. and Cockburn, A.: 1989, *The Fate of the Forest: Developers, Destroyers and Defenders of the Amazon*, Verso.

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.

Leonard, H. S. and Goodman, N.: 1940, The calculus of individuals and its uses, *Journal of Symbolic Logic* **5**, 45–55.

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

Simons, P.: 1987, *Parts: A Study In Ontology*, Clarendon Press, Oxford.

Tarski, A.: 1956, Foundations of the geometry of solids, *Logic, Semantics, Metamathematics*, Oxford Clarendon Press, chapter 2. trans. J.H. Woodger.

Recognition and Diagnosis

Scaling up goal recognition

Neal Lesh and Oren Etzioni*
Department of Computer Science and Engineering
University of Washington, Seattle, WA 98195
neal@cs.washington.edu, etzioni@cs.washington.edu
(206) 616-1849 FAX: (206) 543-2969

Abstract

Traditionally, plan and goal recognition has investigated examples in which the number of plans and goals is less than one hundred. In this paper, we address three challenges to scaling up goal recognition. First, hand coding the plan library is a tedious if not impractical method of generating massive libraries. Second, previous representations for plan libraries are exponentially large in the number of goal predicates. Third, existing algorithms consider each candidate goal which is prohibitively slow. Our solution to the first problem is to automatically construct plan libraries with *biases*, explicit assumptions about the type of plans and goals people have. Our solution to the second problem is to represent the consistent goals with a compact, *version space* representation. Our solution to the third problem is a goal recognition algorithm that can quickly update this compact representation given an observed action. Experimentally, in both software and kitchen domains, we show our prototype implementation to be fast on libraries containing thousands of goals. Analytically, we identify a class of goals for which our algorithm is logarithmic in the number of goals.

1 Motivation

Plan recognition is often studied in domains in which people's behavior is characterized by fewer than 100 plans and goals, such as the traffic [Pynadath and Wellman, 1995] and student-scheduling [Carberry, 1990] domains. Our objective is to apply goal recognition, a special case of plan recognition, to domains in which there are tens of thousands of possible goals and hundreds of plans per goal.

The key factor in scaling up plan or goal recognition is the size of the plan library, which contains the plans and goals the actor might have. Massive plan libraries occur because the library can be exponential in the number of primitive actions and goal predicates in the domain. Just as many sentences can be formed from a few words, many plans can be formed out of a few actions and many goals can be formed out of a few predicates. We address three challenges to recognition:

1. How can we generate the plan library?
2. How can we represent plan libraries compactly?
3. How can we perform fast recognition?

In previous work, the libraries were hand coded. For each new domain, a human expert must describe all plans and goals the observed actor might have. Since hand coding libraries with tens of thousands of goals is an enormous knowledge acquisition task, it is desirable to automate construction of the plan library.

Constructing the library, however, is a delicate task. Automating construction requires articulating the principles that govern which plans and goals belong in the plan library. [Weida and Litman, 1992] and [van Beek, 1996], among others, state that the library must be complete, i.e. it must include every plan and goal the actor might execute. We show that the quality of recognition depends on the content of the plan library in another, more subtle way. In particular, the presence of extraneous plans or goals in a complete plan library can increase the number of observations required to identify the actor's plans and goals.

The second issue we address, compact representation, is also particularly desirable for massive libraries because it is not feasible to explicitly store each plan and

*Many thanks to Paul Barton-Davis, Mathias Bauer, Sandra Carberry, Keith Golden, Steve Hanks, Diane Litman, Mike Perkowitz, Tony Weida, Dan Weld, and Mike Williamson for comments and discussion. This research was funded in part by Office of Naval Research grant 92-J-1946, by ARPA / Rome Labs grant F30602-95-1-0024, by a gift from Rockwell International Palo Alto Research, and by National Science Foundation grant IRI-9357772.

goal. The library has been stored in a variety of compact representations including context-free grammars [Vilain, 1990], belief networks [Huber *et al.*, 1994], message passing networks [Lin and Goebel, 1990], and plan-operator graphs [Lesh and Etzioni, 1995]. These representations store plans implicitly but store goals explicitly and contain pairwise links between goals. Thus, the size of these representations grows quadratically in the number of candidate goals which in turn grows exponentially in the number of goal predicates.

Our third challenge, fast recognition, is important because recognizers must respond quickly to a new observation in order to support real-time interaction with people. [Vilain, 1990] and [Lin and Goebel, 1990] imposed restrictions on the plan language to achieve recognition that is linear in the size of an input plan hierarchy $\mathcal{H}$. Since $|\mathcal{H}|$ is quadratic in the number of goals, the recognition algorithms are not sufficiently fast for large domains. Furthermore, plan steps must be totally ordered and cannot be shared or interleaved. This increases $|\mathcal{H}|$ and thus slows recognition considerably. For example, one partially ordered plan of polling five printers to find a free printer corresponds to 120 totally ordered plans. Additionally, since plans for individual goals cannot be interleaved, each candidate *combination* of goals that people might pursue simultaneously must be explicitly stored in $\mathcal{H}$.

We have adapted ideas and techniques from machine learning to meet these three challenges.

2 Overview

Our approach to the above challenges follows from the observation that concept learning (e.g. [Mitchell, 1982, Shavlik and Dietterich, 1990]) is structurally similar to plan recognition. Concept learners select a concept that describes, or fits, a large set of data on the basis of seeing a small portion of that data. Plan recognizers select a plan or goal that describes, or fits, the actor's behavior based on seeing a small portion of that behavior. For example, a concept learner might generalize from "Nixon lied" and "Reagan lied" to "male Republican presidents lie" or "politicians lie". Similarly, a plan recognizer might generalize from observing a user execute the Unix commands `finger smith@alpha` and `finger smith@beta` to the conclusion that the computer user is searching for the machine that Smith is currently logged onto.

The challenges posed above been solved in concept learning. Concept learners do not require an explicitly enumerated concept library but instead receive a set of primitives. Learners construct their own hypothesis space using syntactic *biases*, or rules for composing predicates into hypothesis. We automate construction of the plan library by using plan and goal biases, or assumptions about what *type* of plans and goals people have. We can define, for example, a plan library that contains all optimal plans for goals of length two, or one that contains all plans that achieve goals formed by conjunction over the input predicates.

Concept learners do not explicitly store each candidate hypothesis. We borrowed the version space representation [Mitchell, 1982] of hypothesis spaces from concept learning. We establish a partial-order on goals, using the generality relationship, and store only the boundaries between the consistent and inconsistent regions of the goal space. We identify a special case of conjunctive search goals for which our representation remains a constant size. In a more general case, our experiments show that it remains several orders of magnitude smaller than the set of consistent goals. In one case, boundaries that contains on average only 100 goal combinations represents a set that contains on average 500,000 consistent goal combinations.

We adapted Mitchell's version space algorithm to produce the BOCE goal recognizer. We provide the bias-specific functions the algorithm needs for initializing the boundaries and quickly updating the boundaries given a new observed action. BOCE can infer from the boundaries that some goal is, is part of, or is sufficient for one of the actor's goals. BOCE calls RIGS [Lesh and Etzioni, 1995], to determine the consistency of individual goals. RIGS represents all consistent plans for each consistent goal using consistency graphs, which are based on plan-operator graphs [Etzioni, 1991, Smith and Peot, 1993, Etzioni, 1993]. In the special case mentioned above, BOCE processes observed actions in time that is logarithmic in the number of goals. Experimentally, in two domains, we show BOCE can process a plan library that contains over a million goals in less than 16 cpu-seconds per observed action.

We sacrifice expressiveness to automatically construct the library and enable fast recognition. Automatic construction is not appropriate for all domains. Hand coded libraries can be finely tuned to the actors' behavior. Our approach will work well if the actors' behavior is sufficiently structured so that fine tuning is unnecessary. Additionally, in very large domains, automatic construction may be the only viable option.

The rest of this paper is organized as follows. We define our terms and the I/O of recognition in section 3, automatic construction in section 4, our representation in section 5, and our algorithm in section 6. We discuss an example in section 7, the scope of our approach in section 8, our experiments in section 9, and related work and conclusions in sections 10 and 11.

3 Definitions

We now introduce our terms and specify the input/output of both plan and goal recognition. Our formulation can accommodate many planning languages, e.g. STRIPS [Fikes and Nilsson, 1971]. The

language must define the concepts of a predicate being *in* a goal, an action being *in* a plan, one action coming *before* another in a plan, an action being an *instance* of an action schema, one plan being a *subplan* of another, and for one goal to *entail* another. Informally, goal g_i entails goal g_j if every state in which g_i is true, g_j is also true. An example goal is (ISA printer x) $\wedge$ (FREE x), where x is a free variable, printer is a constant, and ISA and FREE are the predicates.

Informally, the actor constructs a plan for each of her goals. The plans might share steps. The actor executes her plans, possibly interleaving them, which produces a series of observable actions. Formally, let $\mathcal{GS} = \{g_1, ..., g_n\}$ be the set of goals the actor currently has and $\mathcal{PS} = \{p_1, ..., p_n\}$ be the set of plans the actor is executing: plan p_i is executed for goal g_i. A goal is *necessary* (for one of the actor's goals) if entailed by a goal in $\mathcal{GS}$, and *sufficient* if it entails a goal in $\mathcal{GS}$. A plan p is *necessary* if p is a subplan of a plan in $\mathcal{PS}$, and *sufficient* if a plan in $\mathcal{PS}$ is a subplan of p.

An *actor model* is a tuple $\langle \Lambda, \Omega, K \rangle$ where Λ is a set of action schemas, Ω is a set of predicates, and K is an integer. Informally, the actor is assumed to construct as many as K plans and goals out of the primitives in Λ and Ω, respectively (see assumptions A3 and A4 below). As shown in figure 1, a *plan recognizer* is a function which takes an action sequence $\mathcal{A}$ and an actor model $\langle \Lambda, \Omega, K \rangle$ as input and outputs Nec(x) or Suff(x), where x is a plan or goal, indicating x is necessary or sufficient. A *goal recognizer* is a special case of a plan recognizer that outputs only necessary goals and sufficient goals.

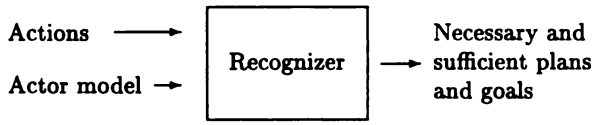


Figure 1: I/O for biased plan recognition

In section 4.2 we show how the recognizer constructs a plan library from the primitives Λ and Ω. The plan library $\mathcal{PL}$ is a set of plan goal pairs $\langle p_i, g_i \rangle$. Each pair $\langle p_i, g_i \rangle \in \mathcal{PL}$ indicates the actor might execute plan p_i for goal g_i. The plan library is *complete* if the actor never executes plan p_i for goal g_i unless $\langle p_i, g_i \rangle \in \mathcal{PL}$.

The recognizer reasons about which combinations of up to K goals in $\mathcal{PL}$ could account for the observed actions $\mathcal{A}$. Action sequence $\mathcal{A} = a_1, ..a_n$ is a *partial interleaved execution* of plans $p_1, .., p_m$ only if (1) for all $1 \leq i \leq n$ there exists some $1 \leq j \leq m$ such that a_i is in p_j and (2) for all $1 \leq i < k \leq n$ and $1 \leq j \leq m$, if a_i and a_k are both in plan p_j then a_i is before a_k in plan p_j. A set of goals $g_1, ..., g_m$ is *consistent* with action sequence $\mathcal{A}$ if there exists plans $p_1, ..., p_m$ such that for all $1 \leq i \leq m$, $\langle p_i, g_i \rangle \in \mathcal{PL}$ and $\mathcal{A}$ is a partial, interleaved execution of plans $p_1, ..., p_m$.

We make the following assumptions:

A1. Action sequence $\mathcal{A}$ is a partial, interleaved execution of the plans $\mathcal{PS}$ the actor is executing.

A2. $\mathcal{PL}$ is complete.

A3. $1 \leq |\mathcal{PS}| = |\mathcal{GS}| \leq K$.

A4. Every action in every plan in $\mathcal{PS}$ is an instance of an action schema in Λ, and every predicate in every goal in $\mathcal{GS}$ is in Ω.

Assumption A1 is important because we cannot recognize an actor's plans from her actions if those actions are not part of her plans. Assumption A2 is important because identifying an actor's plans or goals is essentially the process of eliminating *all* alternative hypotheses, which requires the complete set of possible hypotheses. Assumption A3 is important because, without it, every action might be part of a different plan. Only by observing multiple actions from the same plan can we narrow down the actor's possible plans or goals. Assumption A4 allows the recognizer to construct $\mathcal{PL}$ out of primitives in sets Λ and Ω.

4 Automated construction

We now discuss the content of the plan library and show how to construct a library with biases.

4.1 Extraneous plans and goals

It is often stated (e.g. [Kautz, 1987, Weida and Litman, 1992]) that the plan library must contain every plan and goal the actor could have. It might, then, be tempting to include every conceivable plan and goal in the library. In this section, we show that in addition to being complete, the library should also exclude extraneous plans and goals. A plan or goal is *extraneous* if the actor never executes the plan or has the goal.

First, we review why the plan library should be complete. If the library is incomplete, then the recognizer will produce false conclusions. For example, consider a simple plan library - a cookbook. Suppose we know Smith is cooking one recipe from *The Best Of Lord Krishna's Cuisine* cookbook. After observing Smith slice okra and prepare ghee we conclude that Smith must be cooking Okra Supreme, the only recipe in the *The Best of Lord Krishna's Cuisine* that calls for ghee and sliced okra. Unbeknownst to us, however, Smith is actually using the complete *Lord Krishna's Cuisine* cookbook which includes 8 recipes that call for sliced okra and ghee. Smith's actual meal, Okra Delight, is not in *The Best Of Lord Krishna's Cuisine.*

Now consider the reverse case: we think Smith is cooking from *Lord Krishna's Cuisine* but in fact he is using *The Best Of Lord Krishna's Cuisine.* Our plan library is complete but includes extraneous plans. Again suppose we observe Smith slice okra and prepare ghee.

With the right cookbook, we would know he must be cooking Okra Supreme. But, as it is, there are 8 possible explanations in the cookbook we *think* Smith is using. Thus, the existence of extraneous recipe's prevents us from identifying Smith's goal.

4.2 Bias

Bias fills the gap between the information given to the recognizer (primitives Λ and Ω) and the information needed to perform recognition (a complete plan library $\mathcal{PL}$). *Biases* are grammars for composing primitive actions and predicates into plans and goals. Formally, a *goal bias* GB is a function from a set of predicates to a set of goals; $GB(\Omega)$ refers to the set of goals the actor might compose from the set of predicates Ω. A plan bias PB indicates what plans the actor might construct out of a given set of actions for a given goal. Thus, $PB(g, \Lambda)$ refers to the plans the actor might construct for goal g out of instances of the action schemas in Λ. The plan recognizer's plan library $\mathcal{PL} = \{\langle p, g\rangle \mid g \in GB(\Omega) \wedge p \in PB(g, \Lambda)\}$.

In section 4.1, we showed that recognition suffers if $\mathcal{PL}$ excludes plans and goals the actor does have or includes plans and goals the actor never has. For this reason, the biases should not over-constrain or under-constrain $\mathcal{PL}$. For example, consider the plan bias that admits only optimal plans into $\mathcal{PL}$. If people execute suboptimal plans then this bias over-constrains the library. As described above, this incomplete library will lead to erroneous conclusions. The recognizer might conclude that the observed actor is eating alone when it observes her carry one plate into the dining room, when in fact the actor's actual yet suboptimal plan is to make multiple trips to the kitchen. Conversely, consider the goal bias that allows all conjunctions and disjunctions of goals in Ω. This would under-constrain $\mathcal{PL}$. The observed actions $a_1, .., a_n$ would be consistent with all goals of the form $g_1 \vee ... \vee g_n$ where g_i is an effect of a_i or even an affect of some action a_j where a_i is part of a plan for executing a_j. There would be too many consistent explanations for any given sequence of actions, and the recognizer would never attribute goals to the actor. We currently use the following biases:

Definition 1 (Directed weak) *The* directed weak plan bias *maps Λ and g to the set of all plans p_i s.t. (1) all actions in p_i are instances of action schemas in Λ, (2) every action in p_i supports another action in p_i or supports g and (3) p_i achieves g if executed in some world state s.*[1]

Definition 2 (Conjunctive search) *The* conjunctive search goal bias *maps Ω to the set of all goals, that consist of predicates in Ω, of finding an object that has a conjunction of properties.*

The directed-weak plan bias is flexible in that it allows multiple actions to support the same precondition and allows plans that do not necessarily achieve their goal. Directed-weak plans can contain redundant actions, such as executing a `pwd` command after a `cd` command. Since it allows redundant sensory actions, the directed-weak plan bias does not assume that people remember everything they learn from executing information-gathering commands. This flexibility is important for recognizing the behavior of people. For example, in the Unix domain, we have observed that people often execute long sequences of actions when shorter actions would have sufficed (such as a sequence of `ls`'s and `cd`'s rather than a `find`), and people occasionally give up on tasks that are solvable.

The conjunctive-search goal bias covers an important set of goals because they tend to require long plans. Helping people with long plans is more useful than helping them with something they could do quickly by themselves. Of course, this goal bias only includes some of the goals people might actually have. As described in section 6.2, goal biases can be easily combined together. Thus, we can address various classes of goals separately and combine the solutions together to cover a wide range of behavior. We further discuss these biases in section 7.

5 Compact representation

We now describe our implicit representation for plan libraries which is well suited for fast goal recognition.

We represent only the goals or combinations of goals that are consistent (defined in section 3) with the actions observed so far. We borrow Mitchell's term *version space* to refer to the consistent combination of goals. The version space shrinks as actions are observed and more goals become inconsistent. For example, the goal of finding a free printer would be rejected if we observed `compress foo.tex`.

The initial version space $\mathcal{VS}(\emptyset)$ contains all combinations of up to K goals in $\mathcal{PL}$. Recall that the integer K is an input in the actor model which indicates the maximum number of goals the actor might have. Since a conjunction of goals is itself a goal, for convenience we equate having goals $\{g_1, ..., g_n\}$ with having a single conjunctive goal $g_1 \wedge g_2 ... \wedge g_n$. Formally, $\mathcal{VS}(\emptyset) = \{g_1 \wedge ... \wedge g_n \mid (n \leq K) \wedge (\forall_{1 \leq i \leq n} \exists p, \langle p, g_i\rangle \in \mathcal{PL})\}$. $\mathcal{VS}(\mathcal{A})$ contains the goals consistent with $\mathcal{A}$. Formally, $\mathcal{VS}(\mathcal{A} \neq \emptyset) = \{g \in \mathcal{VS}(\emptyset) \mid g \text{ is consistent with } \mathcal{A}\}$. An important lemma that follows from our assumptions is that $\mathcal{VS}(\mathcal{A})$ will contain the actor's current goals.

Lemma 1 *The actor's current goals $\mathcal{GS} \in \mathcal{VS}(\mathcal{A})$.*

Proof sketch: *Let $\mathcal{GS} = g_1, ..., g_n$ be the actor's goals. By assumption A1, the observed actions $\mathcal{A}$ will be the partial, interleaved execution of plans $p_1, ..., p_n$ where the actor executes plan p_i for goal g_i. By assump-*

[1] The terms in this definition are defined formally in [Lesh and Etzioni, 1995].

tion A2, since the actor executes plan p_i for goal g_i, $\langle p_i, g_i \rangle \in \mathcal{PL}$. By the definition of consistency, these two statements entail that $\mathcal{GS}$ will be consistent with $\mathcal{A}$. By assumptions A2 and A3 and the definition of version spaces, $\mathcal{GS} \in \mathcal{VS}(\emptyset)$. By the definition of version spaces, since $\mathcal{GS} \in \mathcal{VS}(\emptyset)$ and $\mathcal{GS}$ is consistent with $\mathcal{A}$, $\mathcal{GS} \in \mathcal{VS}(\mathcal{A})$.

We order goals by the generality relationship. If every plan for g_i is a plan or subplan for g_j, then g_j is *more general than* g_i, written $g_i \preceq g_j$. For example, "mow the lawn" $\preceq$ "mow the lawn and pay the bills" and "capture Bad Bill, alive" $\preceq$ "capture Bad Bill". Note that these two examples taken together show that generality is different than entailment. Formally, $g_i \preceq g_j$ if every action sequence consistent with g_i is also consistent with g_j. If $g_i \preceq g_j$ and $g_j \not\preceq g_i$ then $g_i \prec g_j$. This closely resembles the definition of generality between plans in [Weida and Litman, 1992].

We represent $\mathcal{VS}(\mathcal{A})$ with a three-tuple $\langle \mathcal{S}, \mathcal{G}, \mathcal{A} \rangle$, where $\mathcal{S}$ is the set of most specific, consistent goals and $\mathcal{G}$ is the set of most general, consistent goals. A goal g_1 is *between* goals g_0 and g_2 iff $g_0 \prec g_1 \prec g_2$. The pair $\langle \mathcal{S}, \mathcal{G} \rangle$ *bounds* $\mathcal{VS}$ iff for every goal $g_i \in \mathcal{VS}$, either $g_i \in \mathcal{S} \cup \mathcal{G}$ or there is a goal $g_s \in \mathcal{S}$ and goal $g_g \in \mathcal{G}$ such that g_i is between g_s and g_g.

We use consistency graphs [Lesh and Etzioni, 1995] to represent the consistent plans for each goal in $\mathcal{S}$ and $\mathcal{G}$. The RIGS algorithm takes a goal set $\mathcal{GS}$, Λ, and $\mathcal{A}$, and constructs and manipulates a consistency graph in order to identify which goals in $\mathcal{GS}$ are inconsistent with $\mathcal{A}$. RIGS runs in time that is linear in $|\mathcal{GS}|$ and polynomial in $|\Lambda| + |\mathcal{A}|$.

- BOCE-initialize(Ω)
  1. Let $\langle \mathcal{S}_0, \mathcal{G}_0 \rangle = \mathit{Init}(\Omega)$
  2. Let $\mathcal{A}_0 = \emptyset$
- BOCE-observe(a_n, Λ, Ω)
  1. Let $A_n = \mathcal{A}_{n-1} + a_n$ (the sequence A_{n-1} with a_n appended to the end)
  2. Let $\langle \mathcal{S}_n, \mathcal{G}_n \rangle = \langle \mathcal{S}_{n-1}, \mathcal{G}_{n-1} \rangle$
  3. FOR every goal $g_i \in \mathcal{G}_n$
     - IF *Consistent*$(g_i, \mathcal{A}_n)$ is `false` THEN remove g_i from $\mathcal{G}_n$
  4. FOR every goal $s_i \in \mathcal{S}_n$
     - IF *Consistent*$(s_i, \mathcal{A}_n)$ is `false` THEN
       (a) remove s_i from $\mathcal{S}_n$
       (b) add to $\mathcal{S}_n$ all goals m_i in *MinGen*$(s_i, \mathcal{A}_n)$ such that *Consistent*$(s_i, \mathcal{A}_n)$ is `true` and *Compare*(s_i, g_j) is `true` for some $g_j \in \mathcal{G}_n$
- BOCE(Ω, Λ)
  1. BOCE-initialize(Ω)
  2. As each action a_n is observed
     (a) BOCE-observe(a_n, Λ, Ω)
     (b) IF $\mathcal{G}_n = \mathcal{S}_n = \{g_i\}$ then return Nec(g_i) $\wedge$ Suff(g_i)
     (c) ELSE IF $\mathcal{S}_n = \{g_i\}$ and `PartialSuff` THEN return Suff(g_i)
     (d) ELSE IF $\mathcal{S}_n = \{g_i\}$ and `PartialNec` THEN return Nec(g_i)
     (e) ELSE IF $\mathcal{S}_n$ or $\mathcal{G}_n$ is empty THEN return `fail`
     (f) ELSE return `null` (*indicating that more observations are needed to recognize goal*)

Figure 2: Pseudo code for BOCE

6 Goal recognition algorithm

We now present the BOCE goal recognition algorithm. First we describe a bias-independent algorithm that calls four bias-specific functions. Then we describe the bias-specific functions that BOCE requires for two classes of conjunctive search goals. Finally, we extend BOCE to handle multiple goals and interleaved plans.

6.1 Boce algorithm

The Based-on-Candidate-Elimination (BOCE) goal recognition algorithm, shown in figure 2, closely resembles the Candidate Elimination algorithm's [Mitchell, 1982] update procedure for a positive example. The pseudo code in figure 2 can be viewed as a template for an algorithm. One needs to "plug in" the *Init*, *Compare*, *Consistent*, and *MinGen* functions (see table 1) to instantiate the goal recognizer. These functions depend on the plan and goal bias. The *Consistent* function for the directed-weak plan bias is described in [Lesh and Etzioni, 1995]. In section 6.2, we describe the *Init*, *Compare*, and *MinGen* for two classes of conjunctive search goals.

The following is a brief sketch of how the algorithm works. The initial library is set to $\langle \mathcal{S}_0, \mathcal{G}_0, \mathcal{A}_0 = \emptyset \rangle$ where $\mathcal{S}_0$ and $\mathcal{G}_0$ are the sets returned by *Init*. The update procedure works as follows. Suppose the current library is $\langle \mathcal{S}_n, \mathcal{G}_n, \mathcal{A}_n \rangle$ and we observe action a_{n+1}. First, BOCE adds a_{n+1} to the end of sequence $\mathcal{A}_n$ to produce $\mathcal{A}_{n+1}$. Then BOCE uses *Consistent* to determine which goals in $\mathcal{S}_n$ and $\mathcal{G}_n$ are inconsistent with $\mathcal{A}_{n+1}$. BOCE removes inconsistent goals from $\mathcal{G}_n$ to produce $\mathcal{G}_{n+1}$ and uses *MinGen* to replace inconsistent goals in $\mathcal{S}_n$ with their minimal generalizations to produce $\mathcal{S}_{n+1}$. BOCE uses *Compare* to remove goals from $\mathcal{S}_{n+1}$ that are more general than other goals in $\mathcal{S}_{n+1}$ or less general than any goal removed from $\mathcal{G}_n$.

The $\mathcal{S}$ and $\mathcal{G}$ boundaries produced by BOCE always describe the version space of consistent goals.

Theorem 1 (Correct boundaries)
Let $\langle \mathcal{S}_n, \mathcal{G}_n, \mathcal{A}_n \rangle$ be the library that BOCE produces after observing actions $a_1, a_2, ..., a_n$. Pair $\langle \mathcal{S}_n, \mathcal{G}_n \rangle$ bounds $\mathcal{VS}(a_1, a_2, ..., a_n)$.

Proof sketch: *This is a proof sketch by induction. The base case is that $\langle S_0, G_0 \rangle$ bounds $VS(\emptyset)$. The induction step is that if $\langle S_{n-1}, G_{n-1} \rangle$ bounds $VS(a_1, ..., a_{n-1})$ then $\langle S_n, G_n \rangle$ bounds $VS(a_1, ..., a_n)$.*

The base case holds because BOCE initializes the boundaries by calling the Init *function. As defined in table 1,* Init *returns a pair that bounds $VS(\emptyset)$.*

By definition, $VS(a_1...a_n) \subseteq VS(a_1...a_{n-1})$. Thus, since $\langle S_{n-1}, G_{n-1} \rangle$ bounds $VS(a_1, ..., a_{n-1})$, step 2 in BOCE-observe establishes that $\langle S_n, G_n \rangle$ bounds $VS(a_1, ..., a_n)$. We now show that steps 3 and 4 preserve this relationship. Step 3 removes inconsistent goals from G_n. Removing g_g from G can only remove goals that are less general than g_g from the space bounded by $\langle S_n, G_n \rangle$. By the definition of generality, a goal that is less general than an inconsistent goal is, itself, inconsistent. If step 4a removes any goal g_i from S_n, then step 4b replaces g_i with its minimal generalizations. Thus the only goal removed from the space bounded by $\langle S_n, G_n \rangle$ by step 4 is g_i itself, which must be inconsistent or else step 4 would not remove it. Thus, steps 3 and 4 only remove inconsistent goals from the space bounded by $\langle S_n, G_n \rangle$. Since $VS(A)$ contains only consistent goals, $\langle S_n, G_n \rangle$ bounds $VS(A)$.

Once the S and G boundaries are computed, we can use them to perform both full and partial recognition. As shown in step 2b of the BOCE function in figure 2, BOCE reports g is both sufficient and necessary for the actor's goals if S and G ever converge to g.

Theorem 2 (Full recognition) *If pair $\langle \{g\}, \{g\} \rangle$ bounds $VS(A)$ then g is necessary and sufficient.*

Proof sketch: *Since $\langle \{g\}, \{g\} \rangle$ bounds $VS(A)$, then $VS(A) = \{g\}$. Recall that g is actually a combination of goals. By lemma 1, the actor's goal set $GS \in VS(A)$. Thus $g = GS$. Thus each goal in g is both necessary and sufficient, by the definition of necessary and sufficient goals.*

Although we do not formalize it here, it is also possible to recognize one of the actor's goals without recognizing all of the actor's goals. If, for example, $\langle \{g_1 \wedge g_2\}, \{g_1 \wedge g_3\} \rangle$ bounds the version space then g_1 must be one of the actor's goals.

The actor's goals can be partially described if generality between goals has a fixed correlation to entailment between goals. The boolean `PartialSuff` is set to true only if for all goals g_i, g_j in $\mathcal{PL}$, g_i entails g_j if $g_i \preceq g_j$. Below, we define a set of goals for which this is true. If `PartialSuff` is true, then in step 2c BOCE indicates that goal g_i is sufficient if the S boundary ever converges to g_i. Similarly, the boolean `PartialNec` is set to true only if for all goals g_i, g_j in $\mathcal{PL}$, g_j entails g_i if $g_i \preceq g_j$. And if `PartialNec` is true then in step 2d BOCE outputs that goal g_i is necessary for one of the actor's goals if the S boundary ever converges to g_i.

function	definition
Init(Ω)	$\langle S, G \rangle$ that bounds $VS(\emptyset)$
Consistent(g, A, Λ)	*true* iff g is consistent with A
Compare(g_i, g_j)	*true* iff $g_i \prec g_j$
MinGen(g, A)	MG st. $\forall g_i \in VS(A)$, $(g \not\prec g_i) \vee (\exists g_j (g_j \in MG) \wedge (g_j \preceq g_i))$

Table 1: BOCE's bias-specific functions.

Theorem 3 (Partial recognition) *If pair $\langle \{g\}, G \rangle$ bounds $VS(A)$ and $g_i \prec g_j$ implies g_i entails g_j, then g is sufficient. If $\langle \{g\}, G \rangle$ bounds $VS(A)$ and $g_i \prec g_j$ implies g_j entails g_i, then g is necessary.*

Proof sketch: *Since $\langle \{g\}, G \rangle$ bounds $VS(A)$, $\{g\}$ is more specific than every other goal in $VS(A)$. Thus, by lemma 1, $\{g\} \preceq GS$ where GS is the actor's goal. If $\{g\}$ is more specific than the actor's goal, and $g_i \prec g_j$ implies g_i entails g_j, then $\{g\}$ must entail GS. By the definition of sufficiency, if $\{g\}$ entails GS then g is sufficient. Similarly, if $\{g\}$ is more specific than the actor's goal and $g_i \prec g_j$ implies g_j entails g_i, then GS must entail $\{g\}$. By the definition of necessary, if the actor's goal GS entails $\{g\}$, then $\{g\}$ is necessary.*

If $S = \emptyset$ or $G = \emptyset$ then one of BOCE's assumptions has been violated, and BOCE returns `fail` at step 2e.

Theorem 4 (Failure) *If $\langle S, \emptyset \rangle$ or $\langle \emptyset, G \rangle$ bounds $VS(A)$ then assumptions 1, 2, 3, and 4 cannot all hold.*

Proof sketch: *If $S = \emptyset$ or $G = \emptyset$ then, by the definition of bounds, $VS(A) = \emptyset$. By lemma 1, $GS \in VS(A)$ and thus $VS(A) \neq \emptyset$. Since lemma 1 follows from assumptions 1, 2, 3, and 4 if S or G is empty, then all of our assumptions cannot hold.*

Finally, in step 2f BOCE outputs `null` if none of the above recognition conditions occur. The output `null` simply means "no comment at this time" which is always true. Thus, the above theorems show that BOCE's outputs are always true.

6.2 Goal bias specific functions

In this section, we discuss two classes of goals: decomposable and composable goals. Decomposable and composable goal sets are mutually exclusive but not exhaustive (some goal sets are neither composable or decomposable). Table 2 describes the *Init*, *Compare*, and *MinGen* functions for these two classes of goals.

We discuss decomposable goals first. Goals are decomposable if adding a conjunct makes them more specific. For example, adding the constraint that the printer can print color to the goal of finding a free printer forms a more specific goal. Formally, goal set GS is *decomposable* if for all $g_i, g_j \in GS$, $g_j \subset g_i$ iff $g_i \prec g_j$.

A conjunctive goal g_i is a subset of g_j iff every conjunct of g_i is also a conjunct of g_j.

Since we only consider conjunctive goals in this section, we use sets to represent goals. The set $\{c_1..c_n\}$ refers to the goal $c_1 \wedge ... \wedge c_n$. In table 2, we represent sets of goals with a set of literals $\mathcal{I}$ and a set of literals $\mathcal{C}$. The literals in $\mathcal{I}$ describe the invariant part of the goals that appears in every goal. For example, in the set of goals of looking for a free printer with some subset of possible printer attributes, the invariant part is "the printer must be free". The set $\mathcal{C}$ describes possible constraints on the goal, such as "the printer must print color". We use $\mathcal{I} \times \mathcal{C}^*$ to refer to the set of goals $\{\mathcal{I} \cup C_i \mid C_i \subseteq \mathcal{C}\}$.

By the definition of the decomposable goal sets, the *Compare* function for decomposable goals is, as shown in table 2, the inverse of the subset function. The *Init* function for decomposable goal set $\mathcal{I} \times \mathcal{C}^*$ is $\langle\{\mathcal{I} \cup \mathcal{C}\}, \{\mathcal{I}\}\rangle$. The $\mathcal{S}$ set contains the most constrained and thus the most specific goal. The $\mathcal{G}$ set contains the least constrained and thus most general goal. This definition of *Init* satisfies BOCE's requirements for the *Init* function, as specified in table 1.

Theorem 5 (Init Decomposable) *If $\mathcal{I} \times \mathcal{C}^*$ is decomposable, then $\langle\{\mathcal{I} \cup \mathcal{C}\}, \{\mathcal{I}\}\rangle$ bounds $\mathcal{I} \times \mathcal{C}^*$.*

Proof: *Let g be a goal in $\mathcal{I} \times \mathcal{C}^*$. Goal g must be of the form $\mathcal{I} \cup C_i$ for some $C_i \subseteq \mathcal{C}$. By the definition of decomposable goal sets, since $\mathcal{I} \subseteq (\mathcal{I} \cup C_i) \subseteq (\mathcal{I} \cup \mathcal{C})$, $(\mathcal{I} \cup \mathcal{C}) \preceq (\mathcal{I} \cup C_i) \preceq \mathcal{I}$. By the definition of bounds, since every goal g_i in $\mathcal{I} \times \mathcal{C}^*$ is between $\mathcal{I} \cup \mathcal{C}$ and $\mathcal{I}$, $\langle\{\mathcal{I} \cup \mathcal{C}\}, \{\mathcal{I}\}\rangle$ bounds $\mathcal{I} \times \mathcal{C}^*$.*

We compute the minimal generalizations of goal $\mathcal{I} \cup C_i$ as follows. For all $c_j \in C_i$, we call *Consistent* to determine if $\mathcal{I} + c_j$ is consistent with $\mathcal{A}$. If $\mathcal{I} + c_j$ is inconsistent, we remove it from C_i. The resulting goal, $\mathcal{I} \cup C_i'$, is the sole generalization of $\mathcal{I} \cup C_i$. This definition of MinGen, shown in table 2, satisfies BOCE's requirements for the MinGen function, shown in table 1.

Theorem 6 (MinGen Decomposable) *If $\mathcal{I} \times \mathcal{C}^*$ is decomposable and $C_i \subseteq \mathcal{C}$, then the minimal generalizations of $\mathcal{I} \cup C_i$ in $\mathcal{I} \times \mathcal{C}^*$ is $\{\mathcal{I} \cup C_i'\}$ where C_i' is the set of all literals c_j in C_i such that $\mathcal{I} + c_j$ is consistent with the observed actions $\mathcal{A}$.*

Proof sketch: *To show $\{\mathcal{I} \cup C_i'\}$ is the minimal generalizations of $\mathcal{I} \cup C_i$, we show that for all goals $g_k \in \mathcal{VS}(\mathcal{A})$ either [1] $(\mathcal{I} \cup C_i) \not\preceq g_k$ or [2] $(\mathcal{I} \cup C_i') \preceq g_k$.*

The version space $\mathcal{VS}(\mathcal{A})$ contains all goals in $\mathcal{I} \times \mathcal{C}^*$ that are consistent with $\mathcal{A}$. Since $g_k \in \mathcal{I} \times \mathcal{C}^*$, $g_k = \mathcal{I} \cup C_l$ for some $C_l \subseteq \mathcal{C}$. Note that $C_i' \subseteq C_i \subseteq \mathcal{C}$. By the definition of decomposable goals, if $C_l \subseteq C_i'$ then $(\mathcal{I} \cup C_i') \preceq g_k$ and condition [2] is met. By the definition of decomposable goals, if $C_l \not\subseteq C_i$ then $(\mathcal{I} \cup C_i) \not\preceq g_k$ and condition [1] is met. If neither $C_l \subseteq C_i'$ nor $C_l \not\subseteq C_i$, then there exists some c_m such that $(c_m \in C_l) \wedge (c_m \in C_i) \wedge (c_m \notin C_i')$. By the way C_i' is constructed, if c_m is in C_i but not C_i' then $(\mathcal{I} + c_m)$ must be inconsistent with the observed actions. Since $c_m \in C_l$, $(\mathcal{I} + c_m) \subseteq g_k$. By the definition of decomposable goals, $g_k \preceq (\mathcal{I} + c_m)$. By the definition of generality, if $g_k \preceq (\mathcal{I} + c_m)$ and $(\mathcal{I} + c_m)$ is inconsistent, then g_k is also inconsistent. Thus $g_k \notin \mathcal{VS}(\mathcal{A})$. Thus, for all $g_k \in \mathcal{VS}(\mathcal{A})$ conditions 1 or 2 above is met.

function	decomposable goal set $\mathcal{I} \times \mathcal{C}^*$	composable goal set $\mathcal{I} \times \mathcal{C}^*$
Init()	$\langle\{\mathcal{I} \cup \mathcal{C}\}, \{\mathcal{I}\}\rangle$	$\langle\{\mathcal{I}\}, \{\mathcal{I} \cup \mathcal{C}\}\rangle$
Compare(g_i, g_j)	*true* iff $g_i \supset g_j$	*true* iff $g_i \subset g_j$
MinGen(g)	Let $g = \mathcal{I} \cup C_i$. Return $\{\mathcal{I} \cup C_i'\}$ s.t. $\forall c_j \in C_i$, Cons($\mathcal{I}$+$c_j, \mathcal{A}$)	Let $g = \mathcal{I} \cup C_i$. Return $\{\mathcal{I} \cup C_i + c_j \mid c_j \in (\mathcal{C} - C_i)\}$
`PartialSuff`	True	False
`PartialNec`	False	True

Table 2: Bias-specific functions for composable and decomposable goals. 'Cons' is shorthand for the 'Consistent' function.

BOCE processes exponentially large decomposable goals in linear time. This is possible because the $\mathcal{S}$ and $\mathcal{G}$ sets always contain at most one goal. Both sets start out as singletons. In general, BOCE can increase the size of $\mathcal{S}$ by replacing a goal with its minimal generalizations. The minimal generalizations of a decomposable goal, however, is always a set containing exactly one goal. Furthermore, the minimal generalizations of a decomposable goal can be computed by checking the consistency of at most $|\mathcal{C}|$ goals. The total number of consistency checks for a single observed actions is, then, $|\mathcal{C}| + 2$ including the two checks of the goals in $\mathcal{S}$ and $\mathcal{G}$. This is logarithmic time because $\mathcal{I} \times \mathcal{C}^*$ contains $2^{|\mathcal{C}|}$ goals.

Theorem 7 (Logarithmic time decomposable) *If the goals in $\mathcal{PL}$ are the decomposable goal set $\mathcal{I} \times \mathcal{C}^*$, then BOCE will compute $\langle\mathcal{S}_{n+1}, \mathcal{G}_{n+1}, \mathcal{A}_{n+1}\rangle$ given $\langle\mathcal{S}_n, \mathcal{G}_n, \mathcal{A}_n\rangle$ and action a_{n+1} by calling the consistency function on at most $|\mathcal{C}| + 2$ goals.*

Proof sketch: *As shown in table 2, for decomposable goal sets, $|\mathcal{G}_0| = |\mathcal{S}_0| = 1$. Since BOCE never adds to the $\mathcal{G}$ set, for all n, $|\mathcal{G}_n| \leq 1$. BOCE can only increase $|\mathcal{S}|$ by replacing a goal in $\mathcal{S}$ with its minimal generalizations. Since, as shown in table 2, the minimal generalizations of any decomposable goal is always a single goal, for all n, $|\mathcal{S}_n| \leq 1$.*

BOCE checks the consistency of goals only in steps 3 and 4 of BOCE-update. In step 3, BOCE-update checks the consistency of $|\mathcal{G}_n| = 1$ goal. In step 4, BOCE-update checks the consistency of $|\mathcal{S}_n| = 1$ goal and calls MinGen *on every inconsistent goal. The MinGen($\mathcal{I} \cup C_i$) needs to determine if goal $(\mathcal{I} + c_j)$*

is consistent for all $c_j \in C_i$. This requires exactly $|C_i|$ calls to the consistency function. Note that $C_i \subset C$. Thus the total calls in steps 3 and 4 of BOCE-update is at most $|C|+2$. No where else does BOCE check the consistency of a goal.

The `PartialSuff` flag can be set to `true` for decomposable goal sets, indicating that if $g_i \prec g_j$ then g_i entails g_j. This is true because if g_i and g_j are decomposable and $g_i \prec g_j$ then $g_j \subset g_i$; if $g_j \subset g_i$ then g_j holds for all worlds in which g_i is true and thus g_i entails g_j. For example, consider the goal of finding a free, color, second-floor printer, or g_{fc2} = (ISA printer x) $\wedge$ (FREE x) $\wedge$ (COLOR x) $\wedge$ (FLOOR x 2). A subset of this goal is g_{fc} = (ISA printer x) $\wedge$ (FREE x) $\wedge$ (COLOR x), the goal of finding a free, color printer. Goal g_{fc2} is sufficient for goal g_{fc}, and, more generally, for any of its subsets.

Now we discuss composable goals. Goals are composable if adding a conjunct makes them more general. For example, if A and B are independent conjuncts then "do A" $\prec$ "do A and B" because anything we observe in service of "do A" could also be in service of "do A and B". Formally, goal set $\mathcal{G}$ is *composable* if for all $g_i, g_j \in \mathcal{G}$, $g_i \subset g_j$ iff $g_i \prec g_j$.

By the definition of the composable goal sets, the *Compare* function for decomposable goals is, as shown in table 2, the subset function. The *Init* function for decomposable goal set $\mathcal{I} \times \mathcal{C}^*$ is $\langle\{\mathcal{I}\}, \{\mathcal{I} \cup \mathcal{C}\}\rangle$. This definition of *Init* satisfies BOCE's requirements for the *Init* function, as specified in table 1.

Theorem 8 (Init Composable) *If $\mathcal{I} \times \mathcal{C}^*$ is composable, then $\langle\{\mathcal{I}\}, \{\mathcal{I} \cup \mathcal{C}\}\rangle$ bounds $\mathcal{I} \times \mathcal{C}^*$.*

The proof is very similar to the proof of theorem 5. For composable goal sets, adding a constraint makes the goal more general. Thus, the minimal generalizations of a composable goal are simply all those goals with exactly one more constraint. Formally,

Theorem 9 (MinGen Composable) *If $\mathcal{I} \times \mathcal{C}^*$ is decomposable and $C_i \subseteq \mathcal{C}$, then the minimal generalizations of $\mathcal{I} \cup C_i$ in $\mathcal{I} \times \mathcal{C}^*$ is $\{\mathcal{I} \cup C_i + c_j \mid c_j \in (\mathcal{C} - C_i)\}$*

Proof sketch: *Let $g_k \in \mathcal{VS}(\mathcal{A})$. The version space $\mathcal{VS}(\mathcal{A})$ contains all goals in $\mathcal{I} \times \mathcal{C}^*$ that are consistent with $\mathcal{A}$. Since $g_k \in \mathcal{I} \times \mathcal{C}^*$, $g_k = \mathcal{I} \cup C_l$ for some $C_l \subseteq \mathcal{C}$. If $C_i \not\subset C_l$, then by the definition of composable goals $(\mathcal{I} \cup C_i) \not\prec g_k$. If $C_i \subset C_l$, then there must be some goal $g_j \in \mathcal{MG} = \{\mathcal{I} \cup C_i + c_j \mid c_j \in (\mathcal{C} - C_i)\}$ such that $g_j \subseteq g_k$. By the definition of composable goals, $g_j \preceq g_k$. Thus, since either $C_i \not\subset C_l$ or $C_i \subset C_l$, for all $g_k \in \mathcal{VS}(\mathcal{A})$, either $(\mathcal{I} \cup C_i) \not\prec g_k$ or $\exists g_j (g_j \in \mathcal{MG}) \wedge (g_j \prec g_k)$. Thus, by the definition in table 1, $\mathcal{MG}$ is the minimal generalizations of $(\mathcal{I} \cup C_i)$.*

We cannot process composable goal sets as efficiently as decomposable ones. In the worst case, *MinGen*(g) produces $|\Omega|$ goals and thus the S and $\mathcal{G}$ boundaries can grow to size $2^{|\Omega|-1}$, half the total number of goals. In section 9 we show that, in practice, the boundaries for composable goals remain small.

The `PartialNec` flag can be set to `true` for composable goal sets, indicating that if $g_i \prec g_j$ then g_j entails g_i. This is true because if g_i and g_j are composable and if $g_i \prec g_j$ then $g_i \subset g_j$; if $g_i \subset g_j$ then g_i holds for all worlds in which g_j holds and thus g_j entails g_i.

Finally, note that our approach is modular. From the definition of bounds, it follows that if $\langle S_1, \mathcal{G}_1\rangle$ bounds version space $\mathcal{VS}_1$ and $\langle S_2, \mathcal{G}_2\rangle$ bounds $\mathcal{VS}_2$ then $\langle S_1 \cup S_2, \mathcal{G}_1 \cup \mathcal{G}_2\rangle$ bounds version space $\mathcal{VS}_1 \cup \mathcal{VS}_2$. Thus, we can run BOCE separately with different sets of bias-specific functions for different goal sets and easily merge the results to produce the boundaries of the union of the goal sets.

6.3 Interleaved plans

We now show how to perform recognition on interleaved plans, given the ability to perform recognition on uninterleaved plans. Suppose we have *Consistent*, *Compare*, *Init*, and *MinGen* for $K = 1$. The objective is construct *Consistent*^{K}, *Compare*^{K}, *Init*^{K}, and *MinGen*^{K} for $K > 1$. Recall that we equate having goals g_1 and g_2 with having goal $g_1 \wedge g_2$. A simple but slow approach would simply be to represent all combinations of goals explicitly, but we show that version spaces allow for more tractable representation.

The function *Consistent*^{K} is the same as *Consistent* because $g_1 \wedge \ldots \wedge g_n$ is a goal just like any other goal. In general, it is not possible to construct *Compare*^{K} because it depends on the plan bias. For example, if the plan bias states that people execute optimal plans then the set of plans for goal g_1 and the set of plans for goal g_2 might have nothing in common with the set of plans for $g_1 \wedge g_2$. We make the following assumption, which holds for our directed-weak plan bias:

A5. Let $s_1, .. s_n$ and $g_1, ..., g_m$ all be goals. $(s_1 \wedge s_2 \wedge .. s_n) \preceq (g_1 \wedge g_2 \wedge .. g_m)$ iff there is an onto mapping f such that for all $1 \leq i \leq n$, $s_i \preceq g_{f(i)}$.

Thus, for example, $g_1 \preceq (g_1 \wedge g_2)$ and if $g_1 \preceq g_3$ and $g_2 \preceq g_4$ then $(g_1 \wedge g_2) \preceq (g_3 \wedge g_4)$. Under this assumption, interleaved goals behave very much like composable goals. The *Init*^{K} function returns $\langle S, \mathcal{G}^K\rangle$. It follows from assumption A5 that if $\langle S, \mathcal{G}\rangle$ bounds $\mathcal{VS}(\mathcal{A})$ then $\langle S, \mathcal{G}^K\rangle$ bounds $\mathcal{VS}(\mathcal{A})^K$.

Version spaces are useful for handling interleaved goals because the *MinGen*^{K} function can exploit a rich generality structure amongst combinations of goals. Let $\mathcal{MG}^1$ be the minimal generalizations of goal g_1 and $\mathcal{MG}^2$ be the minimal generalizations of goal g_2. It is true that $\mathcal{MG}^1 \times \mathcal{MG}^2$ is a minimal generalization of

$g_1 \wedge g_2$. However, there is a more compact, minimal generalization of $g_1 \wedge g_2$, defined as follows:

Theorem 10 (MinGen^K) *The minimal generalizations of goal $g_1 \wedge g_2 \wedge ..g_n$ in $\mathcal{GS}^K$ are all goals of the form $MGJ_i \wedge g_1 \wedge ...g_{j-1} \wedge g_{j+1} \wedge ..g_n$, where $MGJ_i \in \text{MinGen}(g_j)$ and, if $n < K$, goals of the form $g_1 \wedge ... \wedge g_n \wedge g_{n+1}$ where $g_{n+1} \in \mathcal{S}_0$ where $\langle \mathcal{S}_0, \mathcal{G}_0 \rangle$ is the pair returned by* Init *that bounds $\mathcal{GS}$.*

Due to space limitations, we omit the proof. If every goal in $\mathcal{GS}$ has Q minimal generalizations then a goal in $\mathcal{GS}^K$ has at most $(Q \times K) + |\mathcal{S}|$ minimal generalizations. This is much better than the Q^K minimal generalizations that occurs from taking the cross product of the individual generalizations of all the goals.

7 Example

The following example illustrates our notion of biases, the I/O of goal recognition, and how our algorithm works. We begin with an informal description of the interactions among a computer user, a task-completion agent, and a goal recognizer. The user's goal is to find a free, color printer.

First, the user checks printer **alpha**, a color, fourth-floor, double-sided printer made by Xerox in 1991. The goal recognizer ignores where and when the printer was made and tells the task-completion agent that finding a free, color, fourth-floor, double-sided printer is *sufficient* for the actor's goal. The agent, however, fails to locate a printer meeting these requirements.

Next, the user checks printer **beta**, a color, second-floor, double-sided printer made by Xerox in 1993. Now, the recognizer reports that a free, color, double-sided printer is sufficient (dropping the fourth-floor requirement). The task-completion agent finds printer **gamma**, a color, third-floor, double-sided printer made by NEC in 1991. The agent recommends **gamma** to the user, which satisfies her goal.

The biases in section 4 allow BOCE to draw these conclusions. Without a conjunctive goal bias it could not exclude, for example, the possibility that the actor was searching for a printer on the second *or* fourth floor, in which case printer **gamma** would be unsuitable. Without a plan bias that excludes irrelevant actions, the recognizer could not even infer that the actor's goals had anything to do with finding a printer.

For the above interactions to occur, the predicate set Ω must include COLOR, FLOOR, and DOUBLE.SIDED. But Ω must exclude YEAR.MADE and MADE.BY. Otherwise, the recognizer might have instructed the task-completion agent to find a printer made by Xerox. Finally, K must be 1. Otherwise, the two observed actions could have been for two separate goals.

We now describe the $\mathcal{S}$ and $\mathcal{G}$ boundaries that BOCE maintains, assuming the printer search goals are decomposable. $\mathcal{G}_0$ = {(ISA printer x) $\wedge$ (FREE x)}, i.e. it contains the goal of finding a free printer. $\mathcal{S}_0$ = {(ISA printer x) $\wedge$ (FREE x) $\wedge$ (FLOOR x F) $\wedge$ (DOUBLE.SIDED x) $\wedge$ (COLOR x)}. The variable F is a special *schema* variable, which represents all constants appropriate for the FLOOR predicate. It is reasonable to view the goal in $\mathcal{S}_0$ as requiring that the printer be on the first floor, on the second floor, on the third floor, etc. After the computer user polls printer **alpha**, $\mathcal{S}_1$ contains {(ISA printer x) $\wedge$ (FREE x) $\wedge$ (FLOOR x 4) $\wedge$ (DOUBLE.SIDED x) $\wedge$ (COLOR x)}. The $\mathcal{G}$ set is unchanged; $\mathcal{G}_1 = \mathcal{G}_0$. After the computer user polls printer **beta**, $\mathcal{S}_2$ contains {(ISA printer x) $\wedge$ (FREE x) $\wedge$ (DOUBLE.SIDED x) $\wedge$ (COLOR x)}. Again, the $\mathcal{G}$ set is unchanged. At each step, the goal in $\mathcal{S}$ is reported to be sufficient for the actor's goal.

8 Discussion of scope

The simplicity of the above example, designed to illustrate our approach, does not reflect the full scope of our work. As the example suggests, our approach is limited to *keyhole* recognition where the observed actor is unaware of or indifferent to the observer. The example should not be construed to suggest we allow only 'single-level" plans in which every action contributes directly to the goal. Our formulation and the RIGS algorithm allow for arbitrarily "deep" plans with subgoals, subgoals of subgoals, and so on.

The above example is also small. Section 9 describes experiments on large libraries in the Unix and Toast [Agre and Horswill, 1992] domains. We have recently tested BOCE, with similar results, on data gathered under Microsoft Windows 95.[2] Since user help systems are a primary application of keyhole recognition, software systems are an important domain. Additionally, conjunctive search goals are likely to occur in any domains in which people search for objects. Furthermore, our approach could easily be extended to *conjunctive set goals*, such as the goal of compressing all files that are large and have been unmodified for a year. Conjunctive set and search goals are important because they give rise to long plans and thus task-completion can be especially helpful. Additionally, our experiments in the Toast domain show that our recognizer can work on reactive agents that do not actually construct plans, but do act planfully.

What about classes of goals other than composable and decomposable goals? To review, BOCE exploits the generality relationship among goals to maintain a compact representation of the candidate goals. Broadly speaking, this approach will work well when (1) the library has a large number of goals, (2) there is a rich generality structure among these goals, and (3) this generality structure can be determined by the

[2] Contact neal@cs.washington.edu for details and data.

predicates	candidate goals	consistent goals per update	goals in $\mathcal{S}$ and $\mathcal{G}$ per update	cpu seconds per update	percent work saved
4	64	33	6.5	0.4	67.5
6	256	129	9.98	0.9	67.8
8	1,024	486	13.53	1.3	48.0
10	4,096	1,778	18.89	1.8	67.3
12	16,384	7,913	34	5.1	63.5
14	65,536	26,871	38.44	7.7	71.1
16	262,144	116,596	55.13	12.5	52.2
18	1,048,576	643,919	98.5	15.255	61.2

Table 3: Results from Unix experiments: BOCE is fast because it enumerates only a fraction of the plan library. The numbers are averages over 100 randomized runs on our algorithm coded in Lisp on a 134 MHZ SGI Indy.

form, rather than the content, of the goals. A case when these conditions arise, besides conjunctive search goals, is when the actor interleaves goals. Condition (1) holds because the number of combinations quickly explodes. Conditions (2) and (3) hold because (under certain assumptions) $g_i \wedge g_j$ is more general than g_i or g_j. In the degenerate case, however, when (2) or (3) does not hold, BOCE reverts to RIGS and each goal's consistency is individually computed; performance thus becomes linear in the number of goals.

9 Empirical validation

In this section we describe our experiments on realistic data in two domains to investigate whether BOCE works quickly and if its output can be useful.

9.1 Unix

We simulate a world in which an actor searches for a variety of objects, such as files and printers. At each time step the actor executes one action from one of her plans. After each action is observed, BOCE can output necessary or sufficient goals. These goals are used to suggest an object to the actor. For example, if the recognizer outputs that a free, color, second-floor printer would satisfy the actor's goal, and such a printer exists in our simulation, then that printer is suggested to the actor. If the suggested object satisfies one of the actor's goals, then the actor ceases to execute any actions in the plan for that goal.

We generate the action sequence $\mathcal{A}$ by interleaving one or two sequences of actions, each of which represents the execution of a plan to find some object. Some of these sequences were gathered from human subjects solving goals in Unix. In all trial runs, we set $K = 2$, and Λ to be 25 action schemas for various Unix commands. We varied Ω to contain between two and nine predicates for describing printers and two to nine predicates for describing files. The printer goals are decomposable, and the file goals are composable. As we add predicates to Ω, the plan library grows exponentially.

Our prototype implementation in Lisp is reasonably fast. As shown in table 3, BOCE performs recognition on a library containing over a million goals in about 16 cpu-seconds per update. These results suggest we could achieve real-time performance if we rewrote BOCE in optimized C code.

Table 3 shows that BOCE is fast because it enumerates only a fraction of the consistent goals. For 14 predicates, BOCE computes the consistency of 39 goals per update, on average. Maintaining an explicit list of consistent goals would require checking an average of 26,871 goals, per update.

The last column in table 3 quantifies the work saved by task-completion using BOCE, compared to letting the actor solve all her own goals. A 30% savings occurs, for example, if the task-completion agent satisfies the actor's goal after the actor executes 7 actions in a 10 action plan. In our experiments, goal recognition leads to well over 50% work saved.

9.2 Toast

Toast [Agre and Horswill, 1992] is a simulated agent that performs cooking tasks. Toast accepts goals such as making omelettes, cleaning dirty dishes, or setting the table. Toast performs a greedy action-selection that, given certain constraints on the domain, allows it to solve its goals without scheduling any future actions. We extended the default domain to include a variety of goals such as making pancakes with blueberries and omelettes with green peppers and onions.

In each trial run, we gave Toast between 1 and 6 goals from a total of 14 possible goals. Additionally, Toast always pursues a background goal of cleaning the kitchen. As BOCE observes Toast execute actions, BOCE both partially and (with more observations) fully identifies Toast's goals. For example, if Toast pours flour into a bowl, breaks an egg into the bowl, and cleans the cherry pitter then BOCE determines that making pancakes with cherries is necessary for one of Toast's goals, though the pancakes may contain other flavors as well.

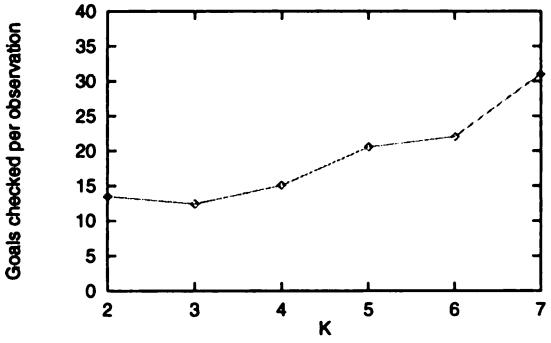


Figure 3: Results from Toast experiments: The size of the $\mathcal{S}$ and $\mathcal{G}$ boundaries grows linearly with K, the number of goals Toast might pursue simultaneously. Since there were 15 individual goals, BOCE implicitly considers 15^K goal combinations. These numbers are the average over 50 randomized runs.

Figure 9.2 shows our version space representation remains small even as the number of goal combinations grows exponentially in K. Since there are 15 total goals, there are 15^K combinations of goals that BOCE must consider. Even for 15^7 goals, the $\mathcal{S}$ and $\mathcal{G}$ sets contain less than 40 goals.[3] The average update time was under 12 cpu-seconds per observation for all tests.

Although we supply the goals, we do not supply the plans. Toast generates the actions to solve its goals. The directed-weak plan bias, described in section 4.2 and computed by the RIGS algorithm, determines which plans are included in BOCE's plan library $\mathcal{PL}$. If the plans that Toast actually executes are not in $\mathcal{PL}$, then BOCE will produce erroneous conclusions. The fact that our recognizer works in this domain reflects the general applicability of the directed-weak bias.

9.3 Comments

These results may not be encouraging because the version space grows exponentially with the number of predicates. One can imagine domains with, say, 200 predicates. These domains will be tractable, however, if the predicates can be factored into subgroups such as ten predicates for describing printers, ten for machines, and so on. In this case, there would be $(20 \times 2^{10})^K$ goals. For comparison, the largest goal set in our Unix experiments was $(2 \times 2^9)^K$, for $K = 2$.

10 Related work

In order to automate construction of the plan library and achieve fast recognition our approach sacrifices expressiveness of the plan representation. Plan hierarchies, meticulously described by [Kautz, 1987], are a widely-used (e.g. [Ardissono and Cohen, 1995, Bauer, 1996, Lin and Goebel, 1990, Vilain, 1990]) example of a more expressive representation: arbitrary plans and goals can be omitted and arbitrary constraints posted between steps of plans. In contrast, our approach requires the plan library to contain all and only plans and goals that share an underlying structure, defined by the plan and goal biases.

Others before us have traded expressiveness for tractability. [Vilain, 1990] converts recognition into context free grammar parsing by requiring plan steps to be totally ordered and not shared or interleaved. The complexity of his algorithm is $O(|\mathcal{H}|^2|\mathcal{A}|^3)$ where $|\mathcal{H}|$ is the size, including edges, of the input plan hierarchy and $\mathcal{A}$ is the observations. [Lin and Goebel, 1990] describe a message-passing scheme with complexity $O(|\mathcal{H}||\mathcal{A}|^3)$ under Vilain's restrictions.

Although the faster of these algorithms is linear in $|\mathcal{H}|$, $\mathcal{H}$ contains every candidate goal. For decomposable and composable goal sets, $\mathcal{H}$ will thus contain $2^{|\Omega|}$ goals, where Ω is the set of predicates for forming goals. In contrast, for decomposable goals, BOCE is linear in $|\Omega|$. Furthermore, our experiments provide reason to believe that BOCE will outperform these algorithms for composable goals as well. We tested BOCE on libraries containing millions[4] of goals. In the above systems, this necessarily leads to grammars with millions of rules or message networks with millions of nodes. In contrast, BOCE does not *have to* enumerate every goal. Our experiments show BOCE's boundaries can contain less than 1/1,000 of all consistent goals.

BOCE's consistency algorithm, RIGS [Lesh and Etzioni, 1995], is polynomial in the number of primitive actions, while Lin and Goebel's algorithm is linear in the number of complex actions in $\mathcal{H}$. The number of complex actions can be exponential in the number of primitive actions. More specifically, the complexity of RIGS is $O((|\Lambda| + |\mathcal{A}|)^6|\mathcal{GS}|)$ where $\mathcal{GS}$ is the set of goals to be tested. BOCE's complexity for decomposable goals is thus $O((|\Lambda|+|\mathcal{A}|)^6|\Omega|)$ and for composable goals is $O((|\Lambda| + |\mathcal{A}|)^6 2^{|\Omega|})$. How does this compare to Lin and Goebel's complexity? For our plan and goal biases, $\mathcal{H}$ will necessarily contain $2^{|\Omega|}$ goals and could contain $2^{|\Lambda|}$ complex actions. Thus $\mathcal{H}$ can contain $(2^{|\Omega|} + 2^{|\Lambda|})^2$ edges, making the above algorithm's worst-case running time $O((2^{|\Omega|} + 2^{|\Lambda|})^2|\mathcal{A}|^3)$.

[Weida and Litman, 1992] automatically compute the generality relationships between plans and also gain considerable efficiency by maintaining only the boundaries within a space of plans. They do, however, require a complete list of all candidate plans as input. In contrast, BOCE constructs its own library and exploits

[3] The background goal has a slighter larger impact on the boundaries than the other goals, which accounts for the drop from $K = 2$ to $K = 3$.

[4] The million pairs of goals in our experiments can be treated as one large goal set. To allow step sharing and interleaving with the above algorithms, each combination of goals needs to be represented explicitly in $\mathcal{H}$

generality to enumerate only a fraction of the library and to enable partial recognition. RIGS is similar in spirit to the Unix Consultant [Wilensky *et al.*, 1988], which also chains actions together. However, they do not construct goals (only plans) or define an implicit plan library, as we do, but instead focus on providing advice to Unix users. [Pynadath and Wellman, 1995] also investigate recognition based on beliefs of how the actor plans. Their work focuses on accounting for the context in which plans are generated and uncertainty in recognition, but does not automatically construct the actor's potential plans or goals. Related to constructing libraries, [Bauer, 1996] learns probabilities for a given hierarchy people's past planning episodes and [Mooney, 1990] learns plan schemata from people's past planning episodes.

11 Conclusions

New techniques are needed to perform plan and goal recognition in domains in which plan libraries contain tens of thousands of goals. Three of our contributions can be summarized as follows.

First, we automate construction of the plan library. Almost all recognizers require a hand coded plan library. Our goal recognizer constructs its own plan library from a much smaller set of primitives.

Second, we extend techniques from machine learning and planning to provide representations and algorithms for fast recognition. New techniques for fast recognition are of interest because recognition is intended to support real-time interaction with people; new techniques are needed because plan libraries can grow exponentially in the size of the domain. We have identified a class of goals for which BOCE is exponentially faster than other fast recognition algorithms.

Third, our experiments represent a substantial advance in empirical evaluation for plan recognition. We test our goal recognizer on large libraries, use data generated independently of the plan library, and quantify the positive impact of goal recognition.

References

[Agre and Horswill, 1992] P. Agre and I. Horswill. Cultural support for improvisation. In *Proc. 10th Nat. Conf. on AI*, pp. 363–368, 1992.

[Ardissono and Cohen, 1995] L. Ardissono and R. Cohen. On the value of user modeling for improving plan recognition. In *Proceedings of the workshop on The Next Generation of Plan Recognition Systems: Challenges for and Insight from Related Areas of AI*, pp. 8–12, August 1995.

[Bauer, 1996] M. Bauer. Acquisition of user preferences for plan recognition. In *Proceedings of the Fifth International Conference on User Modeling*, pp. 105–112, January 1996.

[Carberry, 1990] S. Carberry. Incorporating default inferences into plan recognition. In *Proc. 8th Nat. Conf. on AI*, vol. 1, pp. 471–8, july 1990.

[Etzioni, 1991] Oren Etzioni. STATIC: A problem-space compiler for prodigy. In *Proc. 9th Nat. Conf. on AI*, pp. 533–540, 1991.

[Etzioni, 1993] Oren Etzioni. Acquiring search-control knowledge via static analysis. *Artificial Intelligence*, 62(2):255–302, 1993.

[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(3/4), 1971.

[Huber *et al.*, 1994] J.M. Huber, H.D. Edmund, and M.P. Wellman. The automated mapping of plans for plan recognition. In *Proc. 10th Conf. on Uncertainty in Artifical Intelligence*, pp. 344–51, July 1994.

[Kautz, 1987] H. Kautz. *A Formal Theory Of Plan Recognition.* PhD thesis, University of Rochester, 1987.

[Lesh and Etzioni, 1995] Neal Lesh and Oren Etzioni. A sound and fast goal recognizer. In *Proc. 15th Int. Joint Conf. on AI*, pp. 1704–1710, 1995.

[Lin and Goebel, 1990] D. Lin and R. Goebel. A message passing algorithm for plan recognition. In *Proc. 12th Int. Joint Conf. on AI*, vol. 1, pp. 280–5, July 1990.

[Mitchell, 1982] T. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, March 1982.

[Mooney, 1990] Raymond J. Mooney. Learning Plan Schemata From Observation: Explanation-Based Learning for Plan Recognition. *Cognitive Science*, pp. 483–509, 1990.

[Pynadath and Wellman, 1995] D.V. Pynadath and M.P. Wellman. Accounting for context in plan recognition, with application to traffic monitoring. In *Proc. 11th Conf. on Uncertainty in Artifical Intelligence*, pp. 472–81, August 1995.

[Shavlik and Dietterich, 1990] J. Shavlik and T. Dietterich, eds. *Readings in Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1990.

[Smith and Peot, 1993] D. Smith and M. Peot. Postponing threats in partial-order planning. In *Proc. 11th Nat. Conf. on AI*, pp. 500–506, June 1993.

[van Beek, 1996] P. van Beek. An investigation of probabilistic interpretations of heuristics in plan recognition. In *Proceedings of the Fifth International Conference on User Modeling*, pp. 113–120, January 1996.

[Vilain, 1990] M. Vilain. Getting serious about parsing plans: a grammatical analysis of plan recognition. In *Proc. 8th Nat. Conf. on AI*, pp. 190–197, 1990.

[Weida and Litman, 1992] R. Weida and D. Litman. Terminological Reasoning with Constraint Networks and an Application to Plan Recognition. In *Proc. 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning*, October 1992.

[Wilensky *et al.*, 1988] Robert Wilensky, David Chin, Marc Luria, James Martin, James Mayfield, and Dekai Wu. The Berkeley UNIX Consultant project. *Computational Linguistics*, 14(4):35–84, 1988.

Computing approximate diagnoses by using approximate entailment

Annette ten Teije
SWI
University of Amsterdam
The Netherlands
annette@swi.psy.uva.nl

Frank van Harmelen
Dept. of Math and CS
Vrije Universiteit Amsterdam
The Netherlands
frankh@cs.vu.nl

Abstract

The most widely accepted models of diagnostic reasoning are all phrased in terms of the logical consequence relations. In work in recent years, Schaerf and Cadoli have proposed efficient approximations of the classical consequence relation. The central idea of this paper is to parameterise the notion of diagnosis over different approximations of the consequence relation. This yields a number of different approximations of classical notions of diagnosis. We derive results about the relation between approximate and classical notions of diagnosis. Our results are attractive for a number of reasons. We obtain more flexible notions of diagnosis, which can be adjusted to particular situations. Furthermore, we obtain efficient anytime algorithms for computing both approximate and classical diagnoses.

1 INTRODUCTION

The motivations of this paper come from two areas, namely from the theory of diagnosis and from recent work on approximate entailment. We will discuss each of these in turn.

Diagnosis. The current theories of diagnostic reasoning give a strict definition of the functionality of diagnosis in terms of the classical consequence relation. The current literature gives little or no indication of what to do when the results of diagnostic reasoning do not satisfy the goal of the computation (e.g. when no diagnosis is obtained, or too many, etc.). We propose a number of approximate notions of diagnoses that may be usefully exploited in such cases.

Even in cases where current methods compute good solutions, we may be interested in solutions which are not exact (too large, too few, etc) but cheaper to compute. An advantage of the method we use is that approximate diagnoses can be computed more efficiently than classical diagnoses. In fact, an accumulation of ever better approximate diagnoses is no more expensive to compute than the most precise diagnosis of this series. This can be exploited by iteratively computing ever better approximations. This iteration can stop at a point when either a sufficiently suitable diagnosis has been obtained or when no time for further computation is available, yielding attractive any-time interruptible algorithms.

Approximate entailment. The proposal for approximate deduction from Schaerf and Cadoli (1995) includes a parameter S (a set of predicate letters) which determines both the accuracy and the cost of the approximation. The correct choice of this set S is crucial for the usefulness of the approximation. In the worst case, when choosing S incorrectly, approximate deduction may end up as expensive as classical deduction. In their paper, Schaerf and Cadoli admit that they have very few general strategies for choosing S appropriately. In general of course, this choice must be heuristic, since a perfect choice for S would defeat known complexity boundaries. In this paper, we exploit properties of diagnostic reasoning to propose a number of informed strategies for choosing this crucial parameter.

The structure of this paper is as follows: section 2 summarises the results on approximate deduction from Schaerf and Cadoli. Section 3 repeats some basic definitions on diagnosis and introduces some notation. Section 4 constitutes the main body of this paper. We derive theorems that characterise the influence of approximate entailment relations on the diagnoses that can be obtained. Section 5 discusses ways in which the results from this paper can be put to practical use and section 6 concludes.

2 SUMMARISING APPROXIMATE ENTAILMENT

In this section we will summarise the work in (Schaerf and Cadoli 1995), which defines the approximate entailment relations that we will exploit for our work on diagnoses. Schaerf and Cadoli define two approximations of classical entailment, named $\vdash_1$ and $\vdash_3$ which are either unsound but complete ($\vdash_1$) or sound but incomplete ($\vdash_3$). By analogy, they sometimes write $\vdash_2$ for classical entailment. Both of these approximations are parameterised over a set of predicate letters S (written $\vdash_1^S$ and $\vdash_3^S$) which determines their accuracy. We repeat some of the basic definitions from (Schaerf and Cadoli 1995):

A 1-S-assignment is a truth assignment to all literals such that
- If $x \in S$ then x and $\neg x$ are given opposite truth values
- If $x \notin S$ then x and $\neg x$ are both given the value 0.

A 3-S-assignment is a truth assignment to all literals such that
- If $x \in S$ then x and $\neg x$ are given opposite truth values
- If $x \notin S$ then x and $\neg x$ are *not* both given the value 0.

In other words: for letters in S, these assignments behave as classical truth assignments, while for letters $x \notin S$ they make either all literals false (1-S-assignments) or make one or both of x and $\neg x$ true (3-S-assignments).

The names of the assignments can be explained as follows. In 1-S-assignments means: there is *one* possible model for letters outside S, namely false for both x and $\neg x$. In 2-S-assignments: there are *two* possible models for letters namely x true and $\neg x$ false, or x false and $\neg x$ true. In 3-S-assignments: there are *three* possible models for letters outside S namely x true and $\neg x$ false, or x false and $\neg x$ true or true for both x and $\neg x$.

Satisfaction of a clause by a 1-S- or 3-S-assignment is defined in the usual way: a formula ϕ is satisfied by an interpretation σ if σ evaluates ϕ into true using the standard rules for the connectives.

The notions of 1-S-entailment and 3-S-entailment are now defined in the same way as classical entailment: a theory T 1-S-entails a formula ϕ written ($T \vdash_1^S \phi$) *iff* every 1-S-assignment that satisfies T also satisfies ϕ, and similarly for 3-S-entailment ($T \vdash_3^S \phi$). Similarly, we can speak of 1-S- and 3-S-consistency.

The following syntactic operations[1] can be used to clarify these definitions. For a theory in clausal form, 1-S-entailment corresponds to classical entailment, but after removing from every clause any literals with a letter outside S. When this results in an empty clause, the theory becomes the inconsistent theory $\bot$. Similarly, 3-S-entailment corresponds to classical entailment, but after removing every clause from the theory that contains a literal with a letter outside S. This may result in the empty theory $\top$. These intuitions lead to the main result of (Schaerf and Cadoli 1995):

Theorem 1 (Approximate entailment)

$$\vdash_3^{\emptyset} \Rightarrow \vdash_3^S \Rightarrow \vdash_3^{S'} \Rightarrow \vdash_2 \Rightarrow \vdash_1^{S'} \Rightarrow \vdash_1^S \Rightarrow \vdash_1^{\emptyset}$$

where $S \subseteq S'$. Everywhere in this paper we will use primed letters to represent sets that are a superset of the unprimed letter.

This states that $\vdash_3^S$ is a sound but incomplete approximation of the classical $\vdash_2$. The counterpositive of the second halve of the theorem (reading $\not\vdash_1^S \Rightarrow \not\vdash_1^{S'} \Rightarrow \not\vdash_2$) states that $\not\vdash_1^S$ is a sound but incomplete approximation of $\not\vdash_2$.

Example 1 [Illustrating $\vdash_3^S$ and $\not\vdash_1^S$] We illustrate these notions with the example theory given in figure 1. We shall call this theory BM, for reasons that will become apparent later. We will use this simple theory throughout the paper to illustrate our results.

We can see that $\vdash_3^S$ is incomplete with respect to $\vdash_2$, since in the theory BM of figure 1 we have that classically $BM \cup \{H_3\} \vdash_2 O_1$, but if we restrict S to $\mathrm{LET}(BM) \setminus \{H_3\}$, where $\mathrm{LET}(BM)$ stands for all the predicate letters in BM, we do not have that $BM \cup \{H_3\} \vdash_3^S O_1$. Similarly, $\not\vdash_1^S$ is incomplete with respect to $\not\vdash_2$ since, for example, if $S = \mathrm{LET}(BM) \setminus \{H_0\}$, then $BM \cup \{H_1\} \not\vdash_2 O_1$, but not $BM \cup \{H_1\} \not\vdash_1^S O_1$.

Furthermore, with increasing S, the accuracy of these approximations improves, until the approximate versions coincide with classical entailment when all letters are included in S. Conversely, the approximations trivialise when $S = \emptyset$: any formula is $\vdash_3^{\emptyset}$ satisfiable, and no formula is $\vdash_1^{\emptyset}$ satisfiable.

Without proof, we give a number of simple lemma's on basic properties of $\vdash_i^S$ that we will use in this paper. In the following lemma's, γ and δ are clauses, ϕ is any formula, l is a literal, and x is a proposition letter.

The first lemma shows that a familiar property of $\vdash$ also holds for all approximations:

Lemma 1 (Monotonicity of $\vdash_i^S$)

$$\textit{If} \quad T \vdash_i^S \phi \quad \textit{then} \quad T \wedge x \vdash_i^S \phi$$

[1] It is because of this close correspondence between 1,3-S-entailment and these syntactic operations that we write $\vdash_i^S$ instead of $\models_i^S$, which was the notation used in (Schaerf and Cadoli 1995).

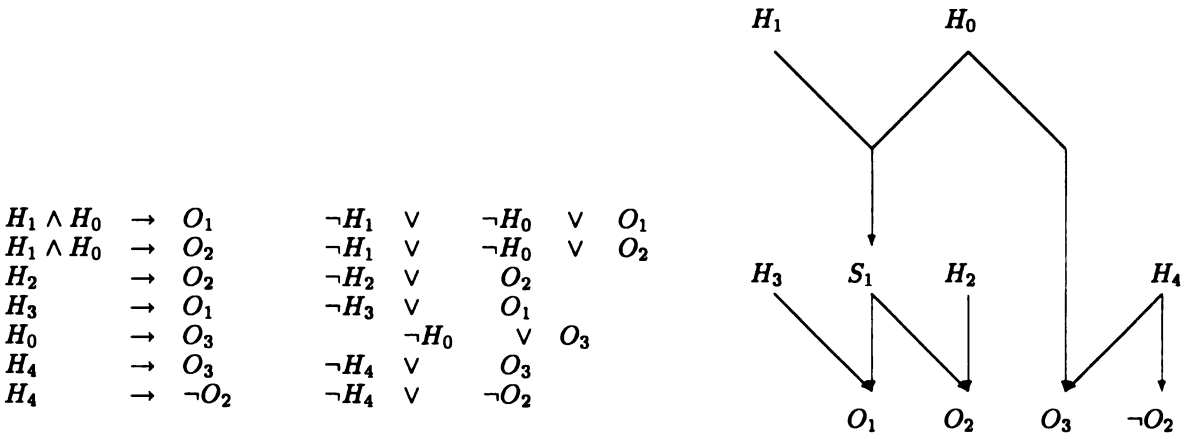


Figure 1: A simple example theory BM (non-clausal, clausal, and as causal network)

The second lemma again shows that a simply property of $\vdash$ continues to hold for $\vdash_i^S$:

Lemma 2 (Extendibility of $\vdash_i^S$)

If $T \not\vdash_1^S \bot$ *and* $x \in S$
then $T \wedge x \not\vdash_1^S \bot$ *or* $T \wedge \neg x \not\vdash_1^S \bot$
If $T \not\vdash_3^S \bot$ *then* $T \wedge x \not\vdash_3^S \bot$ *or* $T \wedge \neg x \not\vdash_3^S \bot$

While the previous lemma said that of a consistent theory at least one extension with x or $\neg x$ remained consistent under the same S, the following lemma says that when moving from S to $S + x$, both extensions remain consistent:

Lemma 3 (Extendability of $\vdash_1^S$ with increasing S)

If $T \not\vdash_1^S \bot$ *then* $T \wedge x \not\vdash_1^{S+x} \bot$ *and* $T \wedge \neg x \not\vdash_1^{S+x} \bot$

The following lemmas are stated informally in (Schaerf and Cadoli 1995). They make precise the syntactic intuitions behind $\vdash_1^S$ and $\vdash_3^S$ discussed above.

Lemma 4 ($\vdash_1^S$ amounts to removing literals)
If $\text{LET}(l) \notin S$ *then*

$$T \wedge (l \vee \delta) \vdash_1^S \gamma \text{ iff } T \wedge \delta \vdash_1^S \gamma$$

Lemma 5 ($\vdash_3^S$ amounts to removing clauses)
If $\text{LET}(l) \notin S$ *and* $\text{LET}(\gamma) \subseteq S$ *then*

$$T \wedge (l \vee \delta) \vdash_3^S \gamma \text{ iff } T \vdash_3^S \gamma$$

Lemma 5.1 in (Schaerf and Cadoli 1995) shows that the condition $\text{LET}(\gamma) \subseteq S$ is no restriction, since $T \vdash_3^S \gamma$ iff $T \vdash_3^{S \cup \text{LET}(\gamma)} \gamma$.

Schaerf and Cadoli also present incremental algorithms for computing $\vdash_1^S$ and $\vdash_3^S$ when S increases. They have obtained attractive complexity results which state that even when computing $\vdash_2$ through iterative computation of $\vdash_3^S$, the total cost of the iterated computation is not larger than the direct computation of $\vdash_2$ (and similarly for $\not\vdash_1^S$ to compute $\not\vdash_2$). However, the iterative computation of the approximate entailment has an important advantage that the iteration may be stopped when a confirming answer is already obtained for a smaller value of S. This yields a potentially drastic reduction of the computational costs. The size of these savings depend on the appropriate choice for S.

In this paper, we use Schaerf and Cadoli's results on approximating the propositional entailment relation. However, in (Schaerf and Cadoli 1995) they show how these results can be extended to first-order theories in a straightforward way: instead of a set of propositional letters, S becomes a set of ground atoms from the Herbrand base.

3 BASIC DEFINITIONS

In this section we introduce the definition of a diagnostic problem and a diagnostic solution that we will use in this paper, and we introduce some notation. We use a common definition of diagnosis that is widespread in the literature. We adopt the mainstream approach of using entailment to characterise the relation between the observations, behavioural model and computed diagnoses. In this paper we present the results on the standard abductive notion of diagnosis. However, we have results on consistency based notion of diagnosis, and results on the combination of abductive based and consistency based notion of diagnosis proposed in (Console and Torasso 1991).

Definition 1 (Diagnostic problem and its solution)
Given a behaviour model BM as a logical theory in

clausal form, and observations O as a set of literals (read as a conjunction), a solution to a diagnostic problem $\langle BM, O\rangle$, is a set of literals E ("E" for explanation, again read as a conjunction), which satisfies the following:

$$BM \cup E \vdash O \quad (1)$$
$$BM \cup E \nvdash \bot \quad (2)$$

These two formulae state that the explanation E must explain the observations O in a non-trivial way.

For technical reasons, we restrict ourselves to causal theories consisting of only two layers. The theory from figure 1 can be interpreted as the causal network from the same figure, where any intermediate layers have been removed in the translation from causal network to theory (in this case the removal of S_1). This restriction does not affect the expressive power of causal theories: any causal theory T containing intermediate layers can be trivially transformed into a theory T' where the intermediate layers have been removed (by resolution on their positive and negative occurrences in the clausal form). T is equivalent to T' in the sense that for any sentence ϕ consisting of only causes and effects, we have

$$T \vdash \phi \text{ iff } T' \vdash \phi$$

A further restriction is that causes in the network may only be positive literals. Observations in the network are allowed to be either positive or negative. Although made for technical reasons, this restriction is observed by many of the causal models found in the diagnostic literature. The work in (Bylander *et al.* 1991) shows that even for such reduced causal theories, the computational complexity of abductive reasoning is sufficiently bad to require the kind of computationally attractive approximations that we investigate in this paper.

Example 2 [Illustrating diagnosis] In figure 1, if we take $O = \{O_1, O_2\}$, then $\{H_0, H_1\}$ and $\{H_0, H_1, \neg H_4\}$ are both diagnoses according to definition 1. The difference between these alternatives is that the the second says that H_4 is false, whereas the first is uncommitted to the states of H_4. A set like $\{H_0, H_1, H_4\}$ is *not* a solution, since it violates (2) via H_4.

Note that we have not mentioned a notion of minimal diagnosis. The literature contains a large number of proposals for different notions of minimality, each of which are appropriate in different contexts. Our theorems are largely independent of particular notions of minimality. We have therefore not included it in our definition, thereby making our results more widely applicable.

The central idea of this paper is to parameterise the notion of diagnosis over different approximations of the entailment relation. In particular, we will want to use $\vdash_1^S$ and $\vdash_3^S$ in our definition of diagnosis. We will write ABD_3^S for formulae (1) and (2) but using $\vdash_3^S$ instead of $\vdash$, and ABD_1^S for using $\vdash_1^S$. We will write ABD_i^S when we intend both ABD_1^S and ABD_3^S. Remember that we write $\vdash_2$ for the classical deduction relation $\vdash$, and we will therefore also write ABD_2 for (1)–(2).

We want to emphasise that our particular definition of a diagnostic problem and its solution, while very common in the literature, is not of crucial importance to our work. It would be well possible to adopt another definition, as long as this definition is again based on logical entailment. The details of our theorems would change, but our central message (namely that approximate entailment can be usefully exploited for diagnostic reasoning to obtain interesting and efficient results) would still hold.

4 APPROXIMATING DIAGNOSES

This section constitutes the main body of work in this paper. We derive theorems that characterise the influence of approximate entailment relations on the diagnoses that can be obtained. The choice for a particular approximation

is characterised by selecting either $\vdash_1^S$, $\vdash_2$ or $\vdash_3^S$, and additionally choosing a value for the set of predicate letters S.

The main intuitions behind using $\vdash_1^S$ and $\vdash_3^S$ in diagnosis are as follows. By using $\vdash_1^S$, candidate solutions more easily satisfy part (1) of our definition of diagnosis, because $\vdash_2 \Rightarrow \vdash_1^S$. Similarly, by using $\vdash_3^S$, candidate solutions more easily satisfy part (2) of our definition of diagnosis, since $\nvdash_2 \Rightarrow \nvdash_3^S$.

Example 3 [Illustrating ABD_1^S and ABD_3^S diagnoses] Again in figure 1, if we take $O = \{O_1, O_2\}$, then $\{H_2, H_3\}$ is an ABD_2 diagnosis, i.e. satisfying (1)–(2). $\{H_2\}$ is not an ABD_2 diagnosis, but if we take $S = \text{LET}(BM) \setminus \{H_3\}$, then $\{H_2\}$ is an ABD_1^S diagnosis. In this case, H_3 is no longer required for explaining O_1. Because $H_3 \notin S$, H_3 and $\neg H_3$ are both false in all 1-S-models, and since $(\neg H_3 \vee O_1) \in BM$, O_1 must be true in all 1-S-models satisfying BM, i.e. $BM \vdash_1^S O_1$. In other words, O_1 requires no further explanation. We will sometimes say that in such a case O_1 is explained "for free".

Similarly, $\{H_0, H_1, H_4\}$ is not a classical ABD_2 diagnosis, since it violates (2). However, if we take $S = \text{LET}(BM) \setminus \{H_4\}$, then $\{H_0, H_1, H_4\}$ is an ABD_3^S diagnosis, showing that it is easier to satisfy (2) through using ABD_3^S.

The organisation of this section is as follows: we start with some observations that restrict the choice of S; we then investigate the effects of using $\vdash_1^S, \vdash_3^S$ in (1–2) (in other words: using ABD_i^S)

4.1 RESTRICTIONS ON CHOOSING S

The first restriction on the choice of S follows from the following theorem. It states that we must demand $O \subseteq S$ if we are to find any diagnoses at all. (Unless stated otherwise, variables are universally quantified):

Theorem 2 (S must contain O)

$$[\exists E : \text{ABD}_1^S(E)] \rightarrow O \subseteq S$$

Proof Suppose that there would be an $o \in O$ with $o \notin S$, then o would have valuation false in all 1-S-models (by definition of $\vdash_1^S$), and then formula (1) can only hold if $BM \cup E$ also has valuation false in all 1-S-models, in other words, $BM \cup E$ would be inconsistent, but this would contradict requirement (2). □

The next theorem states that for ABD_1-diagnoses to exist, at least one letter from each clause in BM must occur in S.

Theorem 3 (S must meet all clauses in BM)

$$[\exists E : \text{ABD}_1^S(E)] \rightarrow \forall \gamma \in BM : \text{LET}(\gamma) \cap S \neq \emptyset$$

Proof If the conclusion of the theorem were not the case, BM would collapse into a 1-S-inconsistent theory, and would never satisfies formula (2). □

Example 4 [Illustrating theorem 3] If we would take $S = \text{LET}(BM) \setminus \{H_3, O_1\}$, we would indeed have a clause with all letters outside S. This would make BM inconsistent, since $(\neg H_3 \vee O_1) \in BM$, but since both H_3 and O_1 are outside S, $\neg H_3$ and O_1 must evaluate to false in all 1-S-models (by definition of $\vdash_1^S$), thereby making BM inconsistent.

Whereas the above two observations restricted S in relation to either O or BM, the following restrict S in relation to the explanation E:

Theorem 4 (S must contain E)

$$\begin{aligned} \text{ABD}_1^S(E) &\rightarrow \text{LET}(E) \subseteq S \\ \text{ABD}_3^S(E) \wedge \subseteq\text{-}min(E) &\rightarrow \text{LET}(E) \subseteq S \end{aligned}$$

where $\subseteq$ -$min(E)$ means that E is a subset-minimal explanation.

Proof For ABD_1, suppose that $x \in E$ and $x \notin S$, then x evaluates to false in all 1-S-models, so E must evaluate to false in all 1-S-models, making $BM \cup E$ inconsistent, thereby violating part (2) of our definition of diagnosis.

For ABD_3^S: suppose there would be an $x \in E$ and $x \notin S$. We will prove that $E \setminus \{x\}$ would then also be an ABD_3^S diagnosis, and this would contradict the assumption that E is subset-minimal. To prove that $\text{ABD}_3^S(E \setminus \{x\})$, we must prove $BM \cup E \setminus \{x\} \not\vdash_3^S \bot$ and $BM \cup E \setminus \{x\} \vdash_3^S O$. The first follows from $BM \cup E \not\vdash_3^S \bot$ (which holds since $\text{ABD}_3^S(E)$) plus using monotonicity (lemma 1), the second follows from $BM \cup E \vdash_3^S O$ (again by $\text{ABD}_3^S(E)$) plus using lemma 5 which applies since $x \notin S$. □

Example 5 [Illustrating theorem 5] For ABD_3^S, take as an example $S = \{H_0, H_1, O_1, O_2\}$, and $O = \{O_1, O_2\}$, then $\{H_0, H_1\}$ is a subset-minimal ABD_3^S diagnosis. Other classical subset-minimal diagnoses (such as $\{H_2, H_3\}$) are not ABD_3^S diagnoses, since their letters are not in S. Non-subset-minimal ABD_3^S diagnoses are allowed to have their letters outside S, for example $\{H_0, H_1, H_3\}$.

On the basis of these initial restrictions on possible choices of S, we are now ready to derive results on the behaviour of ABD_i^S.

4.2 APPROXIMATING ABDUCTIVE DIAGNOSES

In general, a classical abductive diagnosis E can always be arbitrarily expanded to $E \wedge x$ or $E \wedge \neg x$ (this follows directly from lemma's 1 and 2). The following theorem shows that this property (which accounts for the exponential number of abductive diagnoses) continues to hold for ABD_i^S-diagnoses. On the other hand, a classical diagnosis cannot be arbitrarily reduced, since it might get too small to imply O and thereby fail to satisfy formula (1). Surprisingly, ABD_i^S diagnoses *can* always be reduced in a certain way, namely by removing letters not in S. Formally:

Theorem 5 (Changing ABD_i^S diagnoses with fixed S)

$$\begin{aligned} x \in S \wedge \text{ABD}_i^S(E) &\rightarrow \text{ABD}_i^S(E \wedge x) \quad \vee \\ &\qquad \text{ABD}_i^S(E \wedge \neg x) \\ x \notin S \wedge \text{ABD}_i^S(E) &\rightarrow \text{ABD}_i^S(E \setminus \{x, \neg x\}) \end{aligned}$$

Proof First part of the theorem: The condition $x \in S$ is only required for ABD_1^S. For ABD_2 and ABD_3^S, the first part of the theorem already holds without this condition. This is because requirement (1) continues to hold when expanding E (through monotonicity of $\vdash_2$ and $\vdash_3^S$), and requirement (2) continues to hold because at least one of $BM \cup (E \wedge x)$ and $BM \cup (E \wedge \neg x)$ must be consistent if $BM \cup E$ was consistent (extendibility). For $\vdash_1^S$ this extendibility property only holds if $x \in S$.

Second part of the theorem: The second part of the theorem is trivial for ABD_1^S, since, according to theorem 4, $x \notin S$ implies $x \notin E$ in which case $E \setminus \{x, \neg x\}$ simply equals E. The proof for the case ABD_3^S follows from monotonicity (to show consistency of $BM \cup E \setminus \{x, \neg x\}$) and from lemma 5 (to show that $BM \cup E \setminus \{x, \neg x\}$ still implies O). That lemma applies since $x \notin S$. □

Example 6 [Illustrating theorem 5] As an example, we can use the one from example 5, where $\{H_0, H_1, H_3\}$ is an ABD_3^S diagnosis if we take $S = \{H_0, H_1, O_1, O_2\}$, and consequently $\{H_0, H_1\}$ is also an ABD_3^S since $H_3 \notin S$.

Theorem 5 holds for abductive diagnoses under a constant value of S. A related result can be obtained when expanding S with some letter x: when increasing S to $S + x$, for any ABD_1^S-diagnosis E, we can always find a larger ABD_1^{S+x}-diagnosis E' which is a superset of E. Similarly for every ABD_3^S-diagnosis E' there is a smaller ABD_3^{S+x}-diagnosis E with $E \subseteq E'$. As above, such expansion and contraction properties do not in general hold for classical diagnoses.

Theorem 6 (Changing ABD_i^S diagnoses with changing S)

$$\begin{aligned} \text{ABD}_1^S(E) \quad &\rightarrow \quad \text{ABD}_1^{S+x}(E \wedge x) \quad \vee \\ &\qquad \text{ABD}_1^{S+x}(E \wedge \neg x) \\ x \notin S \wedge \text{ABD}_3^S(E) \quad &\rightarrow \quad \text{ABD}_3^{S+x}(E \setminus \{x, \neg x\}) \end{aligned}$$

Proof First part of the theorem: We must prove that the new diagnosis is 1-$S + x$-consistent and 1-$S + x$-implies the observations. Consistency is guaranteed because $BM \cup E$ is 1-S-consistent (since $\text{ABD}_1^S(E)$), and because we extend the existing diagnosis with either x or $\neg x$, lemma 3 says that both extensions will be 1-$S + x$-consistent. Implication of the observations is guaranteed in the following way: First observe that if $x \in S$, implication is guaranteed trivially by lemma 1, so in the following we can assume $x \notin S$. By extending S with x we might loose consequences (observations) that were explained "for free" under $\vdash_1^S$, but these can be recovered by adding x or $\neg x$ to the diagnosis. Since we only have positive literals as causes, it will only ever be needed to add x and never $\neg x$, so it will never happen that we need to add x for one observation and $\neg x$ for another. More formally: Since $\text{ABD}_1^S(E)$, we know that $BM \cup E \vdash_1^S O$, but by extending S with x, $BM \cup E \vdash_1^{S+x} O$ is no longer guaranteed (theorem 1). If $BM \cup E \vdash_1^{S+x} O$ does hold, then so do $BM \cup (E \wedge x) \vdash_1^{S+x} O$ and $BM \cup (E \wedge \neg x) \vdash_1^{S+x} O$ (by monotonicity), and we are done. So lets assume that $BM \cup E \not\vdash_1^{S+x} O$. We know that $BM \cup E \vdash_1^S O$, so the only reason for $BM \cup E \not\vdash_1^{S+x} O$ must be the extra 1-$S + x$-models introduced by allowing x or $\neg x$ to be true (instead of forcing them to be both false because $x \notin S$). Since we only have positive causes in our network, only x can appear as a premise of a causal link and since no node is both a cause ad and effect (since we are in a two-layer network), only $\neg x$ can appear in the clauses of BM. We can therefore remove the additional 1-$S + x$-models for $BM \cup E$ by forcing $\neg x$ to be false. Since $x \in S + x$, this amounts to forcing x to be true, which amounts to adding x to $BM \cup E$. In other words: $BM \cup E \wedge x \vdash_1^{S+x} O$. This proves the first part of the theorem.

Second part of the theorem: We must show that the new diagnosis is 3-$S + x$-consistent and 3-$S + x$-implies the observations. The proof of the latter is easy: The removal of x and $\neg x$ does not lead to loss of implication of any observations, since these literals were not used in the original diagnosis either (because $x \notin S$). More formally: $\text{ABD}_3^S(E)$ means $BM \cup E \vdash_3^S O$, and since $x \notin S$ this is equivalent to $BM \cup E \setminus \{x, \neg x\} \vdash_3^S O$ (by lemma 5), and this in turn implies $BM \cup E \setminus \{x, \neg\} \vdash_3^{S+x} O$ (by theorem 1), as desired. Showing that the new diagnosis is still consistent is somewhat more subtle: by extending S with x we gain additional causal links, namely those links containing either x or $\neg x$ as premise or conclusion. When enabled these additional causal links might cause inconsistency in the following ways: (1) x or $\neg x$ are premises, and the conclusion of the link is inconsistent with others; (2) there are both links with x and links with $\neg x$ as their consequences. However, in case (1) the causal link would never be enabled, since x and $\neg x$ are removed as axioms, and they can also not be implied by other causal links because they are already premises of a causal link, and we are restricted to two-layer causal networks. Case (2) can never happen because such links would already have caused 3-S-inconsistency, which contradicts the condition of the theorem. □

Example 7 [Illustrating theorem 6] To understand this theorem, it is important to realise that ABD_i^{S+x} can be thought of as "more classical than ABD_i^S", in other words, ABD_i^{S+x} is a little bit more like ABD_2 than ABD_i^S was. We should also recall the intuition from the beginning of this section, which stated that ABD_1^S makes it easier to satisfy diagnostic requirement (1) and ABD_3^S makes it easier to satisfy (2).

As an example, we take $S = \text{LET}(BM) \setminus \{H_2\}$, and $O = \{O_1, O_2\}$. Then $\{H_3\}$ is an ABD_1^S diagnosis. When increasing S with H_2, $\{H_3\}$ is by itself no longer an explanation for $\{O_1, O_2\}$, and must be expanded to $\{H_2, H_3\}$, in other words: $\{H_2, H_3\}$ is an $\text{ABD}_1^{S+H_2}$ diagnosis (as predicted by the theorem). Notice that $\{H_3\}$ itself is not an $\text{ABD}_1^{S+H_2}$ diagnosis, since it violates requirement (1): excluding H_2 from S gave us the explanation of O_2 "for free". Conversely, $\{H_2, H_3\}$ is not an ABD_1^S diagnosis.

Similarly, for ABD_3^S if we take $S = \text{LET}(BM) \setminus \{H_4\}$ (again with $O = \{O_1, O_2\}$). Then $\{H_2, H_3, H_4\}$ is an ABD_3^S diagnosis. When adding H_4 to S, $\{H_2, H_3, H_4\}$ is no longer an $\text{ABD}_3^{S+H_4}$ diagnosis, since it violates (2). As predicted by the theorem, $\{H_2, H_3\}$ is an $\text{ABD}_3^{S+H_4}$ diagnosis.

Because the superset and subset relations between diagnoses under different values of S, as in theorem 6, will turn out to be very important, we introduce the following notation:

Definition 2 *For any set P and P', $P \Subset P'$ and $P \Subset + P'$ are defined by:*

$$P \Subset P' \equiv \forall p \in P \quad \exists p' \in P' : \quad p \subseteq p'$$
$$P \Subset + P' \equiv \forall p' \in P' \quad \exists p \in P : \quad p \subseteq p'$$

Example 8 [Examples of $\Subset$ and $\Subset +$]

$$\begin{array}{lcl} & P \Subset P' & \\ \{\{a\}, & \rightarrow & \{\{a,b\}, \\ & \searrow & \\ & & \{a,c\}, \\ \{b,c\}\} & \rightarrow & \{b,c,d\}, \\ & & \{b,e\}\} \end{array} \qquad \begin{array}{lcl} & P \Subset + P' & \\ \{\{a\}, & \leftarrow & \{\{a,b\}, \\ & \nwarrow & \\ & & \{a,c\}, \\ \{b,c\}, & \leftarrow & \{b,c,d\}\} \\ \{e\}\} & & \end{array}$$

injective *surjective*

Notice that $P \subseteq P' \rightarrow P \Subset P'$ and $P \subseteq P' \rightarrow P' \subseteq + P$. We can use this notation to summarise our results so far. If we write $\mathcal{ABD}_i^S$ for the set of all diagnoses E which satisfy $\text{ABD}_i^S(E)$, we have:

Theorem 7 (Relations between $\mathcal{ABD}_i^S$)

$$\begin{aligned} \emptyset = \; & \mathcal{ABD}_1^\emptyset \subseteq \mathcal{ABD}_1^S \Subset \mathcal{ABD}_1^{S'} \Subset \mathcal{ABD}_2 \\ & \mathcal{ABD}_2 \Subset + \mathcal{ABD}_3^{S'} \Subset + \mathcal{ABD}_3^S \Subset + \mathcal{ABD}_3^\emptyset = \emptyset \end{aligned}$$

This states that ABD_1^S diagnoses consist of parts of ABD_2 diagnoses, and that ABD_3^S diagnoses contain ABD_2 diagnoses. **Proof** The theorem is a straightforward reformulation of theorem 6 using the notation of definition 2. □

This theorem only claims that for increasing values of S, we will find superset diagnoses for $\text{ABD}_1^{S'}$ and subset diagnoses for $\text{ABD}_3^{S'}$. The next theorem considerably strengthens this result, by stating that, in the case of ABD_1^S, not only do we find superset diagnoses, but that whenever we find a subset-diagnosis (when increasing S to S'), this $\text{ABD}_1^{S'}$ subset-diagnosis was already an ABD_1^S-diagnosis. Stated differently: when moving from ABD_1^S to $\text{ABD}_1^{S'}$ we will find superset diagnoses, but we will not find any *new* subset diagnoses. The converse also holds: when, while *reducing* S' to S, we find an ABD_1^S diagnoses E' which is a superset of an $\text{ABD}_1^{S'}$ diagnosis E, then this E' was already a diagnosis under $\text{ABD}_1^{S'}$. Roughly speaking, theorem 7 promises that ABD_1^S diagnoses will become larger with increasing S, and the following theorem promises that we will not find any *new* smaller diagnoses when increasing S. A similar result is obtained for ABD_3^S. Formally:

Theorem 8 (No new subset (superset) diagnoses)

Table 1: Summarising the results of section 4.2

ABD_i^S	S	new superset diagnosis	new subset diagnosis	number
$i=1$	bigger	yes	no	more
$i=1$	smaller	no	only	less
$i=3$	bigger	no	yes	more
$i=3$	smaller	only	no	less

$$\begin{aligned} \text{ABD}_1^S(E') \wedge \text{ABD}_1^{S'}(E) \rightarrow \; & \text{ABD}_1^S(E) \wedge \text{ABD}_1^{S'}(E') \\ \text{ABD}_3^S(E) \wedge \text{ABD}_3^{S'}(E') \rightarrow \; & \text{ABD}_3^S(E') \wedge \text{ABD}_3^{S'}(E) \end{aligned}$$

Proof When we write out the definitions for ABD_1^S and $\text{ABD}_1^{S'}$ in the first part of the theorem, we get:

$$\begin{array}{l} BM \cup E' \vdash_1^S O \quad \wedge \quad BM \cup E' \nvdash_1^S \bot \\ \wedge \\ BM \cup E \vdash_1^{S'} O \quad \wedge \quad BM \cup E \nvdash_1^{S'} \bot \\ \rightarrow \\ BM \cup E \vdash_1^S O \quad \wedge \quad BM \cup E \nvdash_1^S \bot \\ \wedge \\ BM \cup E' \vdash_1^{S'} O \quad \wedge \quad BM \cup E' \nvdash_1^{S'} \bot \end{array}$$

$BM \cup E \vdash_1^S O$ and $BM \cup E' \vdash_1^{S'} O$ follow from $BM \cup E \vdash_1^{S'} O$ by theorem 1 and lemma 1 respectively. Similarly, $BM \cup E' \nvdash_1^{S'} \bot$ and $BM \cup E \nvdash_1^S \bot$ follow from $BM \cup E' \nvdash_1^S \bot$ by theorem 1 and lemma 1 respectively.

The proof of the second part of the theorem has the same structure, again using theorem 1 and lemma 1. □

One final result on the relations between approximate abductive diagnoses concerns the sizes of $\mathcal{ABD}_i^S$:

Theorem 9 (Sizes of $\mathcal{ABD}_i^S$)

$$\begin{aligned} 0 = \; & |\mathcal{ABD}_1^\emptyset| \leq |\mathcal{ABD}_1^S| \leq |\mathcal{ABD}_1^{S'}| \leq |\mathcal{ABD}_2| \\ & |\mathcal{ABD}_2| \geq |\mathcal{ABD}_3^{S'}| \geq |\mathcal{ABD}_3^S| \geq |\mathcal{ABD}_3^\emptyset| = 0 \end{aligned}$$

Proof Theorem 6 implies an injective map from $\mathcal{ABD}_1^S$ to $\mathcal{ABD}_1^{S'}$, and from $\mathcal{ABD}_3^S$ to $\mathcal{ABD}_3^{S'}$, as indicated in example 8. This immediately yields the theorem. □

The results of the main theorems in this section are summarised in table 1.

5 HOW TO USE APPROXIMATE DIAGNOSES

We will now briefly discuss some ways in which these results can be exploited for diagnosis. The follow-

Table 2: Behaviour of an anytime algorithm for ABD_1^S for $O = \{O_1, O_2\}$

S	nr	E comment
$\{O_1, O_2\}$	0	*no* S violates theorem 3
$+\{O_3, H_1\}$	0	*no* $H_4 \notin S$, therefore $BM \vdash_1^S \neg O_2$ therefore violation of requirement (2)
$+H_4$	6	$\{\}, \{H_1\}, \{\neg H_1\}, \{\neg H_4\}, \{H_1, \neg H_4\}, \{\neg H_1, \neg H_4\}$ H_2 and H_3 are not in S, therefore $BM \vdash_1^S O_1 \wedge O_2$ (i.e. O_1 and O_2 are explained "for free"), therefore all consistent literal-sets over H_1 and H_4 are diagnoses.
$+H_2$	6	all previous solutions extended with H_2 H_2 is now in S, so O_2 is no longer explained "for free", and therefore H_2 must be included in the solutions.
	4	$\{H_1\}, \{H_1, \neg H_4\}, \{H_1, \neg H_2\}, \{H_1, \neg H_2, \neg H_4\}$ The other explanation for O_2 is H_1 (classically this would be $H_0 \wedge H_1$, but this becomes H_1 since $H_0 \notin S$). We also obtain all consistent extensions of H_1 with letters from S
total	10	
$+H_0$	18	all previous solutions already containing H_2 (6 of them), possibly extended with H_0 or $\neg H_0$ since H_0 is now in S, it is now allowed to include H_0 or $\neg H_0$, but not required, since, $H_3 \notin S$, which still explains O_1 "for free".
	4	$\{H_1, H_0\}, \{H_1, \neg H_4, H_0\}, \{H_1, \neg H_2, H_0\}, \{H_1, \neg H_4, \neg H_2, H_0\}$ these solutions are obtained from those 4 in the previous step where H_1 was required as the explanation for O_2, (i.e. those solutions where H_2 was missing). Since H_0 is now in S, H_1 alone can no longer explain O_2, and we must add H_0.
total	22	
$+H_3$	22	all previous solutions extended with H_3 since H_3 is now in S, it is now allowed to include H_3
	6	$\{H_1, H_2, H_0, \neg H_3\}, \{H_1, \neg H_4, H_2, H_0, \neg H_3\}, \{H_1, H_0, \neg H_3\}, \{H_1, H_0, \neg H_4, \neg H_3\}, \{H_1, \neg H_2, H_0, \neg H_3\}, \{H_1, \neg H_2, H_0, \neg H_4, \neg H_3\}$
	6	$\{H_1, H_2, H_0\}, \{H_1, \neg H_4, H_2, H_0\}, \{H_1, H_0\}, \{H_1, H_0, \neg H_4\}, \{H_1, \neg H_2, H_0\}, \{H_1, \neg H_2, H_0, \neg H_4\}$ In the previous 22 cases, if H_3 is not required for explaining O_1 (because H_0 and H_1 are present), then we are allowed to remove H_3 or to add $\neg H_3$.
total	34	

ing gives us an anytime algorithm (Russel and Zilberstein 1991) for computing diagnoses: start by computing $\mathcal{ABD}_1^S$ for some small value of S, and iteratively increase the value of S. This will include ever more causes in the set of diagnoses $\mathcal{ABD}_1^S$. This set will be an ever better approximation of the set of classical diagnoses $\mathcal{ABD}_2$. The behaviour of this algorithm is described in table 2. This table shows which ABD_1^S solutions are computed when starting with $O = S = \{O_1, O_2\}$, and gradually adding letters to S as indicated. In the final row of the table, $S = \text{LET}(BM)$, and the diagnoses correspond to exactly the set of all 34 classical ABD_2 diagnoses. This algorithm can be interrupted at anytime, and will give monotonically improving results as run-time increases. If for example, starting by including in S only the most urgent causes, we obtain only the urgent parts of ABD_2 diagnoses. When we gradually increase S by adding less urgent causes, we obtain ever larger segments of ABD_2 diagnoses.

The complexity results in (Schaerf and Cadoli 1995) ensure that this converging computation of $\mathcal{ABD}_2$ will not be more expensive than the direct computation of $\mathcal{ABD}_2$ by classical means. However, the results of (Schaerf and Cadoli 1995) that we have applied to abduction concern the worst-case complexity of the declarative characterisation of deduction. Further study and experimentation is needed to see how well these results apply to specific algorithms for abduction known in the literature.

An alternative anytime algorithm is obtained by using ABD_3^S: Again, we start by including in S only the most urgent causes. The $\subseteq$-*min*-ABD_3^S diagnoses are

now exactly those ABD_2 diagnoses which consist of only these urgent causes from S. When increasing S by including less urgent causes, we obtain more and more of $\mathcal{ABD}_2$. Thus, ABD_1^S are the urgent subsets of classical diagnoses, whereas $\subseteq\text{-}min\text{-}\text{ABD}_3^S$ are only those classical diagnoses consisting entirely of urgent causes. Both of these anytime algorithms ensure that we only loose less urgent causes from our approximate diagnoses (theorem 4). Another example of the choice for S would be the most frequently occuring causes, etc.

In (ten Teije and van Harmelen 1996), we have explored some of such properties of the behaviour model BM in order to select predicate letters in S. Some examples from (ten Teije and van Harmelen 1996) are:

- *specificity of observations*: observations are more specific if they have fewer possible causes in the model. For example, in a medical context, headache or a mild fever would be aspecific symptoms, while a lump in the breast would be a very specific symptom. Beginning with the most specific symptoms in S, and gradually adding less specific symptoms, we obtain a decreasing series of diagnoses where inconsistency with specific observables is taken more seriously then inconsistency with a-specific observables. (For this example we used an approximation of a consistency-based notion of diagnosis á la (Reiter 1987), instead of the abductive notion used in this paper).
- *strength of causal connections*: following (Console and Torasso 1990), we distinguish causes which *necessarily* imply their effects from causes which only *possibly* do so, but necessarily. Using ABD_3^S, we first include only necessary causal links in S, and only add possible causal links if no diagnosis can be obtained without them. This results in an algorithm which computes the most reliable diagnoses first, before investigating diagnoses that are based on less reliable causal links.
- *structure in the causal model*: in the context of a simple causal model of an automobile in (ten Teije and van Harmelen 1996), we distinguished sub-models for electrical or mechanical faults. Different sub-models can be gradually incorporated in the diagnostic process by added their letters to S.

6 CONCLUSION AND FUTURE WORK

In this paper, we have extended a widely accepted definition of diagnosis by using approximate deduction relations instead of the usual classical deduction. This has yielded a number of interesting approximate versions of diagnosis. We have proven theorems which state how these approximations can be used to increase and decrease both the total number and the individual size of the computed diagnoses, while guaranteeing certain properties (e.g no loss of classical diagnoses, or only loss of certain types of classical diagnoses). We have exploited our results in efficient anytime algorithms for computing both approximate and classical diagnoses. Finally, we have sketched how the results from this paper can be put to practical use.

As stated above, the results from (Schaerf and Cadoli 1995) ensure that the problem of iterated computation of approximate abduction is not harder (and often easier) than the problem of computing classical abduction, but these are only worst-case results that apply to the declarative presentation of the problem. Further study and experimentation is needed to see how well these results apply to specific algorithms for abduction known in the literature.

Furthermore, in this paper we have studied the existence of diagnoses ($\text{ABD}_i^S(E)$, e.g. theorems 5, 6 and 8), and the set of all diagnoses ($\mathcal{ABD}_i^S$, e.g. theorems 7 and 9). Since the set of all S-diagnoses is exponential (up to 2^S), we will in general not want to compute all of these. Other properties which might be fruitfully studied in future work are: the complexity of finding the next diagnosis, testing whether a formula holds in all or some diagnoses, etc.

In section 3, we stated the restriction to causal networks of only two layers, and we argued that this restriction does not affect the expressiveness of the causal networks. Although this is true, this restriction does affect the different possibilities for approximation: since all intermediate nodes have been removed from the network, these letters can no longer be used to characterise approximations by excluding them from the set S. An example of this is the following simple causal theory T:

$$T = \{H_1 \rightarrow N, H_2 \rightarrow N, N \rightarrow O_1, H_1 \wedge H_2 \rightarrow O_2\}$$

This can be transformed to the two layer theory T':

$$T' = \{H_1 \rightarrow O_1, H_2 \rightarrow O_1, H_1 \wedge H_2 \rightarrow O_2\}$$

Although T and T' are classically equivalent, in the sense that for any sentence ϕ made from the letters H_1, H_2, O_1 and O_2 we have

$$T \vdash \phi \text{ iff } T' \vdash \phi,$$

this is not the case for approximate deduction. In particular, if $S = \text{LET}(T) \setminus \{N\}$,

$$T' \cup \{H_1, H_2\} \vdash_3^S O_2 \wedge O_1$$

while

$$T \cup \{H_1, H_2\} \vdash_3^S O_2 \quad \text{but}$$
$$T \cup \{H_1, H_2\} \nvdash_3^S O_1,$$

This shows that in T', $\{H_1, H_2\}$ is an ABD_3^S-diagnosis for both O_1 and O_2, while in T it is only an ABD_3^S-diagnosis for O_2. In fact, in T' it is impossible to find any choice for S that will yield an ABD_3^S-diagnosis for

O_2 that not also implies O_1, while in T this is possible ($S = \text{LET}(T) \setminus \{N\}$ as above). This shows that two-layer networks really do restrict our options for choosing particular approximations. We are currently investigating under which conditions our results still hold for networks with intermediate layers in networks.

More speculative is the use of our results to model the iterative behaviour of various existing diagnostic systems. Such systems iterate over multiple models (Abu-Hanna 1994), different abstraction levels (Mozetic 1991, Console and Torasso 1992), or request additional observations (Mcilraith and Reiter 1992). Our claim is that such iterative behaviour can be formalised in a uniform way through our results. This would yield insights in the differences and commonalities between such systems, and would make our anytime algorithms available to these existing systems. Further work is required to make these claims more precise.

References

T. Bylander, D. Allemang, M. C. Tanner, and J. R. Josephson. The computational complexity of abduction. *Artificial Intelligence*, 49:25–60, 1991.

L. Console and P. Torasso. Hypothetical reasoning in causal models. *Int. J. of Intelligent Systems*, 5(1):83–124, 1990.

L. Console and P. Torasso. A spectrum of logical definitions of model-based diagnosis. *Computational Intelligence*, 7(3):133–141, 1991.

S. McIlraith and R. Reiter. On tests for hypothetical reasoning. In W. Hamscher, L. Console, and J. de Kleer, editors, *Readings in Model-Based Diagnosis*, pages 89–96. Morgan Kaufman, 1992.

R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–96, 1987.

S. J. Russell and S. Zilberstein. Composing real-time systems. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence, Sydney, Australia*, volume 1, pages 212–217, San Mateo, 1991. Morgan Kaufmann.

M. Schaerf and M. Cadoli. Tractable reasoning via approximation. *Artificial Intelligence*, 74(2):249–310, April 1995.

A. ten Teije and F. van Harmelen. Examples of approximations in diagnosis based on approximate entailment. In *Submitted to DX-96, Seventh International Workshop on Principles of Diagnosis*, October 1996.

Inheritance

Inheriting Well-formed Formulae in a Formula-Augmented Semantic Network

Leora Morgenstern
IBM T.J. Watson Research Center
30 Saw Mill River Road, Hawthorne, NY 10532
leora@watson.ibm.com

Abstract

We examine a commercial application, a large portion of whose domain knowledge is taxonomic in nature, and investigate the suitability of inheritance networks for the task of reasoning about knowledge in this area. We claim that standard inheritance networks are not sufficiently expressive to capture domain knowledge, and introduce *formula-augmented semantic networks* (FANs), a semantic network which attaches well-formed formulae to nodes. We investigate the concept of inheriting wffs in such a network and give a method for computing the sets of well-formed formulae that apply to a given node. Finally, we compare FANs to more familar knowledge structures and the task of inheriting wffs to more familar concepts such as reasoning about sets of default rules in non-monotonic systems.

1 OVERVIEW/MOTIVATION

Inheritance has long been recognized as useful in business domains. This paper investigates the suitability of inheritance networks for reasoning about benefits in the medical insurance domain. The goal has been to develop an expert system that can answer customers' questions about their insurance benefits. Customers typically ask whether a particular service — such as a hip replacement or a CAT scan — is covered, and what restrictions and limitations apply to this coverage (e.g., deductible, maximum number of services per benefit period).

Because so much of the information about medical services is taxonomic in nature — e.g., spinal manipulation is a type of physical therapy; physical therapy, speech therapy, and occupational therapy are all types of therapy — and because coverage and accompanying restrictions are to a large extent inherited along these taxonomic lines, it is natural to encode much of this information in an inheritance network. On the other hand, a significant chunk of information is in the nature of complex business rules specifying certain restrictions that cannot easily be mapped into a network structure. For example, rules such as

> The initial newborn exam must be performed during the mother's hospitalization for delivery

and

> If a patient has exceeded the substance abuse benefits for the year, payment for hospitalization charges may be made under the patient's inpatient hospital benefit

do not map into class-subclass or class-attribute links. Instead, it seems more natural to regard such rules as well-formed formulae, and to reason directly with such wffs.

To accomplish this, we introduce an extension of inheritance networks which we call *formula-augmented semantic networks* or *formula-augmented inheritance networks* (FANs). A FAN looks much like an inheritance network, except that a set of wffs may be associated with each node. The question that we must then answer is: can the wffs at a node be inherited by more specific nodes in the network? General wff-inheritance leads to rampant inconsistency; thus, the main focus of this paper is on how we can perform wff-inheritance in a coherent manner. In particular, we discuss criteria of specificity and explicit path preference in guiding wff-inheritance.

This paper is organized as follows. We describe the commercial domain and the benefits inquiry system. We then define formula-augmented semantic networks and specify the task of wff-inheritance. Subsequently, we explore various methods for wff-inheritance and provide a method for computing the wffs that apply to a node. Finally, we discuss the relation between FANs and other knowledge structures, and between wff-inheritance and reasoning about prioritized rules in other non-monotonic systems.

2 FORMULA-AUGMENTED SEMANTIC NETWORKS

2.1 BENEFITS INQUIRY

The need for formula-augmented semantic networks was discovered while developing a benefits inquiry system for a large medical insurance corporation. The company offers its members a choice of *products*. A product is a collection of benefits, services, and business rules, which are related in the following manner: A service may be *covered* or *excluded* by a particular benefit, and is subject to some set of rules, specifically cost-share, administrative, and medical rules, as well as (possibly) restrictions on the set of medical professionals or facilities that are available. Each product has on the order of 500–1000 rules.

A benefits inquiry system is intended to aid customer service representatives in answering customers' questions. Customers may wish to know if a particular service is covered, or the specific rules that limit coverage. Examples of typical questions are:

> "Will my son's tonsillectomy be covered?"
>
> "How many days can I stay in the hospital after a standard delivery?"

An inheritance hierarchy is a natural choice to represent much of the underlying knowledge for this domain, since, as mentioned in Section 1, so much of the information is taxonomic in nature or is inherited along the taxonomy.

Representing business rules within an inheritance hierarchy is a difficult task. Simple rules such as

> The copay for medical services is 20%

can easily be represented by specifying the attribute of having a 20% copayment. But many of the rules are considerably more complex. For example, consider the rule

> Medical services incurred by an organ donor are covered by the recipient's Human Organ Transplant benefit only if the donor is otherwise not covered.

There does not seem to be any natural way to represent this in a standard inheritance network. This motivates our definition of a formula-augmented semantic network in the next section. In Section 4, we consider whether FANs are truly necessary, or whether we can encode the information in a more traditional structure.

2.2 DEFINITION OF A FAN

A Formula-Augmented Semantic (or Inheritance) Network is a tuple $(\mathcal{N}, \mathcal{W}, \mathcal{B}, \mathcal{L}1, \mathcal{L}2, \mathcal{O})$, where
- $\mathcal{N}$ is a set of nodes,
- $\mathcal{W}$ is a set of wffs,

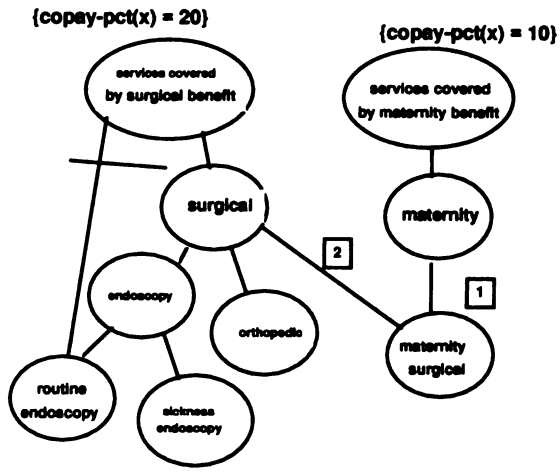


Figure 1: *A portion of the medical insurance network. Lines represent is-a links; slashed lines represent cancels links*

- $\mathcal{B}$ is the background information or background context,
- $\mathcal{L}1$ is a set of links connecting nodes — i.e. $\mathcal{L}1 \subseteq \mathcal{N} \times \mathcal{N}$,
- $\mathcal{L}2$ is a set of links connecting nodes to sets of wffs — i.e. $\mathcal{L}2 \subseteq \mathcal{N} \times 2^{\mathcal{W}}$, and
- $\mathcal{O}$ is a (possibly empty) set of partial orderings on some of the members of $\mathcal{L}1$, subject to certain restrictions which are discussed below.

Intuitively, in this domain, each instance of a FAN represents a particular product of the medical insurance corporation.

The elements of the tuple are described in detail below. The description of $\mathcal{L}1$, which is part of any standard inheritance network, may be safely skipped by those familiar with the work of (Horty et al., 1990). $\mathcal{W}$, $\mathcal{B}$, $\mathcal{L}2$, and $\mathcal{O}$, however, are unique to FANs.

2.2.1 Nodes

A node represents some set of medical services. E.g., the node Surgical represents the set of surgical services; the node X-ray represents the set of x-ray services.

A node may represent a set of services that are covered by a particular benefit. For example, in Figure 1, the node Services Covered by Surgical Benefit represents the set of all medical services that are covered by the Surgical Benefit.

2.2.2 Wffs

The set of wffs $\mathcal{W}$ consists of well-formed formulae of a sorted first-order logic. An example of a rule is

$\forall$ *e1,e2, e3, m, b, i1, i2*
(delivery-event(e1,m,b) $\wedge$ *hospital-event(e2,b)*
$\wedge$ *assoc-hosp-stay(e2,e1)* $\wedge$ *newborn-exam-ev(e3,b)*
$\wedge$ *time(e2,i1)* $\wedge$ *time(e3,i2)* $\wedge\neg$ *subinterval(i2,i1))*
$\supset \neg$ *covered-as(e3,newborn-exam),*
where *e1,e2,* and *e3* are variables ranging over events; *i1* and *i2* are variables ranging over time intervals; and *b* and *m* are variables ranging over babies and mothers, respectively.

This is the first-order translation of the business rule

> The initial newborn exam must be performed during the mother's hospitalization for delivery.

2.2.3 Background Information

The *background* $\mathcal{B}$ is a (possibly empty) set of wffs of first-order logic. Intuitively, $\mathcal{B}$ represents the background information that is true. It includes all rules that are true of all medical services and benefits. [1] For example, the background context may include a wff which states that all medical services must be provided by licensed medical professionals in order to qualify for reimbursement. $\mathcal{B}$ may also include information about particular individuals, such as a patient's medical records and the name of her primary care physician, as well as pricing charts for drugs and pay scales for medical professionals. As these examples make clear, although the information at $\mathcal{B}$ is considered to be global, in the sense that it applies to every node in the network, it may also be subject to frequent change.

2.2.4 Links

Links are relations on objects. There are two sorts of links: links joining nodes and links joining nodes to sets of wffs.

$\mathcal{L}1$: **Links on nodes:** The standard positive and negative is-a links relate nodes. (Negative is-a links will also be referred to as cancels links.) Positive and negative is-a links may be *strict* or *defeasible.* If x and y are nodes, we write x $\Rightarrow$ y and x $\not\Rightarrow$ y to represent, respectively, the strict positive and strict negative is-a links; we write x $\rightarrow$ y and x $\not\rightarrow$ y to represent, respectively, the defeasible positive and defeasible negative is-a links. Intuitively, x $\Rightarrow$ y means that all x's are y's; x $\not\Rightarrow$ y means that all x's are not y's (no x's are y's); x $\rightarrow$ y means that typically, x's are y's; x $\not\rightarrow$ y means that typically, x's are not y's. If x $\Rightarrow$ y or x $\rightarrow$ y, we say that x is a *child* of y or that y is a *parent* of x.

Our account of paths is based on (Horty, 1994). A path is a restricted sequence of positive and/or negative links. We may define paths recursively in the following manner:

- There is a path from x to y iff there is a positive path from x to y or there is a negative path from x to y.
- If x $\rightarrow$ y or x $\Rightarrow$ y (resp., x $\not\rightarrow$ y or x $\not\Rightarrow$ y), then there is a positive (resp. negative) path from x to y.
- If there is a positive path from x to y, and y $\rightarrow$ z or y $\Rightarrow$ z (resp. y $\not\rightarrow$ z or y $\not\Rightarrow$ z), then there is a positive (resp. negative) path from x to z.
- If there is a negative path from y to z and x $\rightarrow$ y or x $\Rightarrow$y, then there is a negative path from x to z.

Note that positive paths can have only positive is-a links; negative paths can have just one negative link at the very end of the path. Paths that contain only strict links are called *strict paths*; paths that contain only defeasible links are *defeasible paths*; paths that contain both strict and defeasible links are called *mixed paths.* For ease of presentation, this paper will be concerned almost exclusively with defeasible paths; extensions to strict and mixed paths are straightforward [2]. The notation $\pi(x, \sigma, y)$ (resp. $\pi'(x, \sigma, y)$) represents a positive (resp. negative) path from x to y through the path σ. We extend this notation so that $\pi(x, \sigma)$ (resp. $\pi'(x, \sigma)$) represents a positive (resp. negative) path from x to the last point of σ going through the path consisting of all but the last point of σ. The function *endpoint*(σ) will be used to designate the last point on the path σ.

Often, there will be both positive and negative paths between two nodes. To determine which path to choose, we follow the analysis of Touretzky(1986) and Horty(1994). Given a *context* — an inheritance network Γ and a set of paths Φ (intuitively, a set of paths arising out of the network), a path (of length greater than 1) is *inheritable* or *undefeated* if it is *constructible* and neither *pre-empted* nor *conflicted.* Briefly, a path is *constructible* in a context if it can, recursively, be built out of the paths in the network; a path is *conflicted* in a context if there is a path of opposite sign in the context with the same starting and ending points; a path is *preempted* if there is a conflicting path with more direct information about the path's endpoint (i.e., a direct link from an earlier point in the path). We also say that a path $\pi(x, \sigma, y)$ is preempted or conflicted if $\pi(x, \sigma)$ is preempted or conflicted. Intuitively, the undefeated or inheritable paths are those which "win out" over their rivals. If there is an undefeated positive (resp. negative) path between x and y, we say x $\rightsquigarrow$ y (resp. x $\not\rightsquigarrow$ y).

Examples of paths can be seen in Figure 1. There is a positive path from Orthopedic Surgery to Services Covered by Surgical Benefit. There are both positive and negative paths from Routine Endoscopy to Services Covered by Surgical Benefit; however, according to Horty et al.'s (1990) specificity criterion (or Touretzky's (1986) inferential distance criterion), the positive path is preempted by the negative path. Thus, the negative path is undefeated.

[1] If we wish, we may think of the wffs in $\mathcal{B}$ as being included in the set of rules at each node.

[2] Note also that in all examples, nodes represent sets of objects, rather than individual objects. This is done to simplify the exposition; the extension to individuals is straightforward.

$\mathcal{O}$: **Ordering on Links:** Multiple inheritance arises when there is an undefeated path from x to y, an undefeated path from x to z, and y $\neq$ z. We call any such point x a *fork point* of the network, and the paths originating from a fork point *multiple paths.* [3] Inheritance networks in the literature have traditionally considered multiple inheritance only when these multiple paths have been initial segments of conflicting paths — as is the case in Figure 1, where the multiple paths π(Routine Endoscopy, Endoscopy) π'(Routine Endoscopy, Services Covered by Surgical Benefit) are initial segments of conflicting paths. We call such cases of multiple inheritance *conflicting-path multiple inheritance.*

In this paper, we will be interested in multiple paths even where they are *not* initial segments of conflicting paths, i.e., *non-conflicting-path multiple inheritance.* Examples of non-conflicting-path multiple inheritance can be seen in Figure 1, where there are distinct non-conflicting paths between Maternity Surgical and Maternity, and between Maternity Surgical and Surgical.

In such cases, we may allow the prioritization of a particular path. We do this by first specifying a partial order on certain links of $\mathcal{L}1$. Specifically, if $x \rightarrow y_1$, $x \rightarrow y_2$, ..., $x \rightarrow y_n$ are elements of $\mathcal{L}1$ and are not initial segments of conflicting paths, we may place a partial order on the $x \rightarrow y_i$'s; this partial order is an element of $\mathcal{O}$.

The partial order on paths can then be defined recursively as follows:
- $x \rightarrow y$ is preferred to $x \rightarrow z$ if $((x,y),(x,z)) \in \mathcal{O}'$, where $\mathcal{O}'$ is an element of $\mathcal{O}$.
- $\pi(x,\sigma,y)$ is preferred to $\pi(x,\tau,z)$ if $\pi(x, endpoint(\sigma))$ is preferred to $\pi(x, endpoint(\tau))$.
- $\pi(x,\sigma,y)$ is preferred to $\pi(x,\sigma,z)$ if $\pi(endpoint(\sigma), y)$ is preferred to $\pi(endpoint(\sigma), z)$.

If there are p fork points in the network, there are at most p partial orders (elements) in $\mathcal{O}$. The number of partial orders in $\mathcal{O}$, and the elements of these partial orders may be further restricted; in particular, we may not wish to specify priorities on links that are initial segments of conflicting paths, or may insist that the priorities be placed in a particular way. We discuss this further in Section 3.2.4.

$\mathcal{L}2$: **Links between nodes and wff sets:** Let N be a node, and W a set of wffs. $N \rightarrow_w W$ means, intuitively, that each w of W is typically true at node N. $N \Rightarrow_w W$ means that each wff w of W is true at N; $N \not\rightarrow_w W$ means that each wff w of W is typically false at node N; $N \not\Rightarrow_w W$ means that each wff w of W is false at N. In practice, we rarely use the $\Rightarrow_w$link. Intuitively, this link is used only when a formula is true at all nodes Ni such that $Ni \rightsquigarrow N$; i.e., we assume that there are no exceptions to N. This happens only rarely.

[3] As opposed to a *branch point*, which is downward.

Since W can be a singleton, we allow the overloading of the $\rightarrow_w$, $\Rightarrow_w$, $\not\rightarrow_w$and $\not\Rightarrow_w$links so that they can refer to individual wffs as well as sets of wffs. If $N \rightarrow_w W$ or $N \Rightarrow_w W$, we will sometimes refer to W as the set of wffs at N or wffs(N). If w is a member of W, we say $at(w, N)$.

The set of rules at each node in a network is constrained to be consistent with the background information $\mathcal{B}$.

2.3 FANs with Added Link Types

The informal description of an insurance product in Section 2.1 might lead one to expect the network to have nodes representing benefits as well as nodes representing sets of services. We would then need covers and excludes links between benefit nodes and service nodes, as well as the usual taxonomic links connecting service nodes to service nodes (and benefit nodes to benefit nodes).

Although this representation is in some sense more natural than then one given in this paper (and is in fact the representation used in the implemented benefits inquiry system), we have opted against it. Because we wish to model this network as much as possible as a standard inheritance network, we choose to reify a node representing a benefit x as a node representing the services covered by benefit x (as in Figure 1). This allows us to use standard is-a and cancels links instead of covers and excludes links. This decision is not integral to the concept of a FAN. We could easily augment the definition of a FAN to include links such as covers and excludes, and specify the interaction between these links and the taxonomic links.

Although we do not explore this option further in this paper, we do note here an interesting feature of the FAN with covers and excludes links (called a modified FAN for the purposes of this discussion) for the insurance application under discussion. The only taxonomic link needed is the standard (strict) is-a link. There is no need for defeasible links, and thus no need for cancel links. Intuitively, the reason for this is that in the insurance application, most of the taxonomic links are genuine and not defeasible subset relations. The only place the defeasible link is needed is between a node and a "services-covered-by-benefit-x" node. But this link is replaced by the covers or excludes nodes in a modified FAN.

Indeed, this discussion highlights an interesting feature of FANs in this domain: cancel links always go directly up to the root node. Due to this feature, the algorithm to identify undefeated paths is much simpler than the Horty et al. (1990) or Stein (1992) algorithm, and in particular, requires only a simple upward traversal from a node until one reaches either a cancels link or a root. While this feature has been exploited in the implementation, this paper discusses the more

general case of FANs in which general inheritance with exceptions is allowed.

It should also be noted that the problem of wff-inheritance discussed in this paper applies not only to FANs but to modified FANs. This shows that the wff-inheritance problem can arise in a strict taxonomy as well as in an inheritance hierarchy with exceptions.

3 INHERITING WELL-FORMED FORMULAE

3.1 THE PROBLEM

One of the defining features of FANs (as opposed to standard semantic networks) is that wffs are associated with nodes. This feature prompts the following question:
What wffs are *inherited* by a node in a FAN? In a standard inheritance network, we focus primarily on the question: is there an undefeated positive path between A and B? Since most nodes in a network represent sets, and since the top node in an inheritance hierarchy often represents some attribute, the most intuitive interpretation of this question is: does some attribute hold of set A? Such questions are of interest in this inheritance network as well: we wish to know whether a particular medical service is covered by some benefit. In addition, however, we are interested in determining which business rules or wffs *apply* or *pertain* to that service (equivalently, to the node representing that service).

Formally, we pose the question as follows. We introduce the following notation: N$\uparrow w$ if wff w pertains or applies to node N. We overload the $\uparrow$ symbol for sets, and say N$\uparrow W$ if $\forall\ w\ w \in W$ iff N$\uparrow w$. We define Ψ(N) = $\{w \mid \text{N}\uparrow w\}$. That is, Ψ(N) is the set of wffs that apply to node N. Then the question is: given a node N in a FAN, what is the set Ψ such that N$\uparrow\Psi$?

A naive approach might suggest taking the union of the wff set attached to N, together with the wff sets attached to all ancestors of N — or more precisely, of the wff sets attached to all those nodes to which there is an undefeated positive path from N. That is,
$\Psi(\text{N}) = \bigcup_{N \rightsquigarrow N_i}$ wffs(N_i)$\cup$ wffs(N).
Such an approach, however, is obviously wrong. Consider, for example, Figure 2. There is an undefeated positive path from N3 to N1. Thus, according to the naive approach,
Ψ(N3) = wffs(N3) $\cup$ wffs(N1) =
{ P $\neg$ Q } $\cup$ { R, P $\supset$ Q }. But this set of wffs is obviously inconsistent. Note also that the naive approach fails to correctly compute Ψ(N2). Although wffs(N1) $\cup$ wffs(N2) is consistent, it is not consistent with respect to the background information $\mathcal{B}$ = { $\neg$ R $\vee\neg$ S }. Clearly, we do not want to blindly take unions of wff sets.

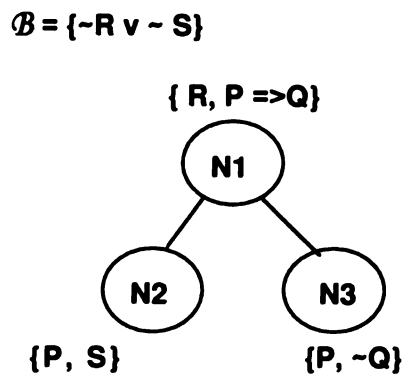


Figure 2: *Taking the union of wffs at nodes yields inconsistency.*

The problem arises because the wffs at a node are usually *typically* true at a node rather than always true at a node. In particular, a set of wffs W may be typically true at a node N, but may not be typically true at all subclasses of N. We discuss ways of dealing with this inconsistency in Section 3.2.

3.2 EFFECTING INHERITANCE OF WFFS

The previous section demonstrated that a naive approach to wff inheritance — namely, taking the union of wff sets at all nodes to which there is a positive undefeated path — leads to inconsistency. A proper approach to wff-inheritance must *recognize* potential inconsistency in inheriting wffs and *resolve* these inconsistencies.

Thus, one could describe the problem of determining which wffs apply to a focus node N as consisting of the following steps:

1. Calculating all the nodes Ni such that N $\rightsquigarrow$ Ni

2. Taking the union of the wffs at N, the wffs at all Ni calculated above, and the background information, and determining if this set is consistent.

3. If this union is not consistent, choosing a maximally consistent subset of wffs(N) $\cup\bigcup_{N \rightsquigarrow Ni}$ wffs(Ni). More precisely, we choose a subset S that satisfies the following conditions:
a. $S \subset$ (wffs(N) $\cup\bigcup_{N \rightsquigarrow Ni}$ wffs(Ni)).
b. $S \cup \mathcal{B}$ is consistent.
c. S is the largest such subset; that is, there does not exist S' satisfying conditions a. and b. such that $S' \subset S$.
We say that S is a maximally consistent subset of wffs(N) $\cup\bigcup_{N \rightsquigarrow Ni}$ wffs(Ni) with respect to (wrt) $\mathcal{B}$. If the meaning is clear, we may omit the reference to $\mathcal{B}$.

We make two brief remarks about the second step:

First, deciding whether a set of rules is consistent is only semi-decidable in general; however, we can restrict our attention to certain subsets of sentences such as those without existential quantifiers or self-embedding function symbols. Such sets are decidable. Second, in decidable cases, deciding whether or not a set of sentences is consistent is intractable; we discuss ways of handling this problem in Section 3.3.

Most of the conceptual difficulties arise in the second task: choosing a maximally consistent subset. In general, there is more than one maximally consistent subset of rules. The question is: which maximally consistent set should we choose? Since each maximally consistent subset is formed by deleting some of the wffs in an inconsistent set, an alternate phrasing of this question is: which wffs should we delete? That is, how do we decide that some wffs are more important than others? The general strategy is to articulate some *preference principles* for sets of wffs and to choose maximally consistent subsets in accordance with these principles.

The particular strategy developed in this paper is to examine preference principles that are based on the structure of the inheritance network. That is, we exploit as much as possible the structure of the inheritance network while we are constructing and choosing maximally consistent subsets. We focus on specificity and multiple paths, both non-conflicting and conflicting. We turn to several examples to illustrate this approach.

Two remarks on these examples: First, some examples deal with paths of length 1; however, we generalise to paths of arbitrary length in the discussion and specification of the wff-inheritance procedure. Second, to simplify and shorten the exposition, we have chosen very simple wffs; the inconsistencies are typically arithmetic in nature. As noted in Sections 1 and 2, rules are typically much more complex, and contradictions between these complex wffs are rampant.

3.2.1 Specificity

Consider the example in Figure 3. HGH (Human Growth Hormone) → R_x Drugs. Suppose we have the following wffs at these nodes: (All variables are assumed to be universally quantified unless otherwise specified.)
Drugs:
{ copay-pct(x) = 10
(Non-network(p) ∧ filled(x,p)) ⊃ penalty(x) = $15}
(There is a 10% copay for all drugs and there is a $15 penalty for prescriptions if they are filled at non-network pharmacies.)
HGH:
{ copay-pct(x) = 50 }
(The copay for HGH drugs is 50%).
If we assume that the background context $\mathcal{B}$ entails

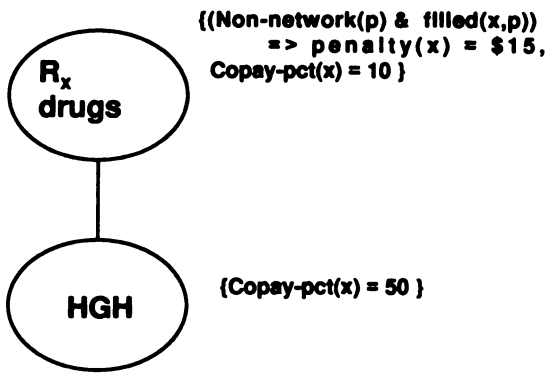


Figure 3: *The wffs at HGH are more specific than the wffs at* R_x *Drugs*

some basic arithmetic facts such as
(v1 ≠ v2) ⊃
(¬ (copay-pct(x) = v1 ∧ copay-pct(x) = v2)) —
that is, a service cannot have two co-pay percentages, then the union of these two sets is inconsistent with respect to $\mathcal{B}$ [4].

Obviously, there are two maximally consistent subsets:
1. The set of wffs at the Drugs node, namely:
{ copay-pct(x) = 10
(Non-network(p) ∧ filled(x,p)) ⊃ penalty(x) = $15 }
and
2. { copay-pct(x) = 50
(Non-network(p) ∧ filled(x,p)) ⊃ penalty(x) = $15 }

The choice of which maximally consistent subset to prefer is clear. The HGH node is more specific than the Drugs node; thus we prefer the subset that has the wff from the HGH node to the subset that has the wff from the Drugs node. That is, we prefer subset 2 to subset 1.

We say that subset 2 is a *preferred maximally consistent subset (pmcs)* relative to the sets of wffs at HGH and wffs at Drugs, with the set of wffs at HGH preferred over the set of wffs at Drugs. We define this concept formally below:

[4]The actual network contains the following cost-share rule at the Prescription Drugs node: copay(x) = $3 (There is a $3 copay for all prescription drugs). The union of the two sets is then not, strictly speaking, inconsistent, as long as HGH drugs cost $6. In other words, the union of these sets entail that HGH drugs cost $6. Detecting unreasonable consequences such as this is an interesting problem in its own right, but beyond the scope of this paper. Presumably, there are several ways to deal with this problem: one can include price tables in the background information, or include general principles forbidding ridiculous consequences in the background information. The difficulty in the latter strategy then is a matter of expressing such principles and ensuring that we have got them all. (I am grateful to Ernie Davis for pointing out this problem.)

Def. (Preferred Maximally Consistent Subset): *Assume that S1, S2 are sets such that S1 ∪ S2 is inconsistent with respect to B. Let X1 and X2 be maximally consistent subsets of S1 ∪ S2 wrt B. Let R1 be a subset of S1; let R2 be a subset of S2, such that R1 = X1 - X2; R2 = X2 - X1. That is, R1 and R2 are all that distinguish X1 from X2. If this is the case, we say that X1 ≅ X2 with respect to R1,R2. Then if S1 is preferred to S2, X1 is a preferred maximally consistent subset of S1 ∪ S2. Otherwise, if S1 ∪ S2 is consistent with respect to B, S1 ∪ S2 is a (in this case* the*) preferred maximally consistent subset of S1 and S2.*

Note that there is not necessarily a unique preferred maximally consistent subset. We introduce the function PMCS where PMCS(S1,S2) returns the *set* of all preferred maximally consistent subsets of S1 and S2, with S1 preferred to S2. By a slight abuse of notation, we will use the notation pmcs(S1,S2) to mean the (random or nondeterministic) function that returns one of the elements of PMCS(S1,S2). In addition, we extend pmcs to n sets, so that pmcs(S1, ..., Sn) = pmcs(pmcs(S1, ..., Sn-1), Sn).

3.2.2 Multiple Paths (Non-conflicting)

Specificity is clearly not the only criterion one can use in determining a preferred maximally consistent subset. Consider again Figure 1. In this figure, the node Maternity Surgical has links to both Maternity and Surgical and thus inherits wffs from both nodes. The union of the wffs at these nodes { copay-pct(x) = 10, copay-pct(x) = 20} is obviously inconsistent with respect to the background constraint mentioned above. There are obviously two maximally consistent subsets:
{copay-pct(x) = 10}
{copay-pct(x) = 20}.
Although specificity does not help here — Maternity is neither more nor less specific than Surgical — it is clear which subset we should prefer. Since the link between Maternity Surgical and Maternity has preference over the link between Maternity Surgical and Surgical, it is reasonable to prefer the first subset, since this keeps the wff of the Maternity node. Thus, one can compute the rules that apply to Maternity by computing pmcs(wffs(Maternity), wffs(Surgical)). In fact, if there were wffs at Maternity Surgical as well, these wffs might contradict either the wffs at Maternity, the wffs at Surgical, or both. Since the wffs at more specific nodes are preferred, the computation of the rules that apply to Maternity Surgical would then be
pmcs(wffs(Maternity Surgical),
pmcs(wffs(Maternity), wffs(Surgical))).

3.2.3 A Procedure for General Paths

The examples discussed thus far have dealt only with paths of length 1. Even in such simple cases, however, one may need to consider wffs at more than 2 nodes; the order in which one takes preferred maximally consistent subsets is obviously crucial. We now discuss this issue for more general paths.

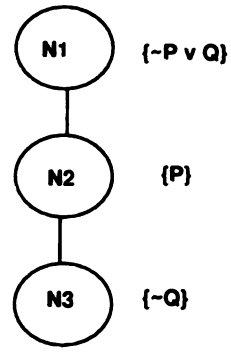


Figure 4: *Doing a simple downward traversal will give the wrong answer for wff-inheritance*

Consider first a node N in a network where there is one path from N to a root node (and thus, no forking points), and consider how to compute Ψ(N). It is clear what we do *not* want to do. We do not want to first take the union of all sets of wffs at the nodes on the path from N to the root, then take maximally consistent subsets of this large set, and finally choose preferred maximally consistent subsets relative to the specificity criterion. It is hard to beat this method for inefficiency. At the very least, the procedure to compute Ψ(N) ought to iteratively traverse the path, computing Ψ(N) as one goes along.

Upward vs. Downward Traversal: The obvious question to ask is whether to traverse the path upward or downward. A downward traversal might seem natural for the following reasons: First, a downward traversal emphasizes the natural analogy between wff-inheritance and (iterated) belief revision (Gardenfors, 1988; Darwiche and Pearl, 1994). Specifically, one might think of inheriting wffs as the process of revising the beliefs at a more general node by the wffs at a more specific node; this translates to the downward traversal of a path in the network. Second, one might expect downward traversal to have the side benefit of computing the sets of wffs that apply to intermediate nodes on the path.

However, straightforward downward traversal — a simple recursive procedure in which one sets Ψ(root) = wffs(root), and for each non-root node x, Ψ(x) = pmcs(wffs(x), Ψ(parent(x)) — turns out to be incorrect. Consider computing Ψ(N3) in Figure 4. It is clear that the expected answer is {P, ¬ Q }; the union of all wff sets is inconsistent, and one discards the wff at N1 since that is the least specific. However, if one traverses the path downward, one gets
Ψ(N1) = {P ⊃ Q},
Ψ(N2) = {P ⊃ Q, P}.

When one reaches N3, one gets
$\Psi(N3) = pmcs(\{\neg Q\}, \{P \supset Q, P\})$.
But there are two preferred maximally consistent subsets: $\{P, \neg Q\}$ and $\{P \supset Q, \neg Q\}$ — contrary to expectations. The problem is that during the downward traversal, we have lost the information that P comes from a node that is more specific than the node from which $P \supset Q$ comes.

There are clearly ways to fix this problem: we can, for example, keep track of the source node of each of the wffs that has been collected so far.

If we wish to avoid this sort of bookkeeping, however, the simplest method for computing $\Psi(N)$ that is consistent with the specificity constraint is based on an upward traversal of the network. Specifically, one begins at the focus node N, taking wffs(N) as the starting set. One then proceeds up the path, at each node taking a preferred maximally consistent subset of the set computed so far and the wffs at the current node. This process will ensure that the specificity constraint is obeyed.

We must also ensure that path ordering is respected in case of forking paths. This is done by examining all links at each point in the path, ordering them, and recursively proceeding up the more preferred links before the less preferred links.

In addition, we must ensure that we do not collect rules from nodes that are only on conflicted or preempted paths. The simplest way to avoid this problem is to preprocess the FAN to remove the preempted and conflicted links. We may do this by using an extension/modification of the procedure given in (Stein, 1992), which computes the *specificity extension* at a focus node.

We thus have the following procedure to compute $\Psi(N)$ in a FAN:

```
PROCEDURE COMPUTE Ψ(N)
Preprocess the FAN to remove
   preempted, conflicted links
Ψ(N) := wffs(N)
FAN-traverse(N)

PROCEDURE FAN-TRAVERSE(x)
if x has not been visited
 mark x as visited
 Ψ(N) := pmcs(Ψ(N),wffs(x))
 If x is a leaf
   then return
   else
    Determine all nodes yi such that x →yi
    Do topological sort of all k links x →yi
    for i = 1 to k do
     FAN-traverse(yi)
```

Note that since we are traversing a dag, as opposed to a tree, some nodes may be visited twice. To avoid recomputing preferred maximally consistent subsets (an expensive operation), we mark nodes as we visit them.

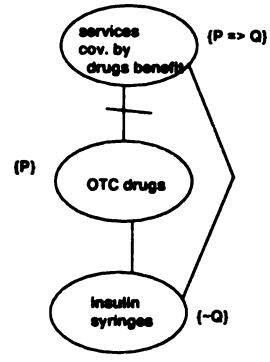


Figure 5: *Which node gets preference? Does preemption make a difference?*

It will be noted that the computation of $\Psi(N)$ does not necessarily produce a unique set. This is because, as noted, the function pmcs does not yield a unique preferred maximally consistent subset. The existence of multiple maximally consistent subsets has always been something of a problem for nonmonotonic theories. In practice, however, the existence of multiple preferred maximally consistent subsets is not at all problematic for this domain. Indeed, it is perfectly legitimate to reason with any preferred maximally consistent subset of wffs that obeys the criteria of specificity and path preference. (In particular, members who are insured by some product often "win a case" by demonstrating that there is a coherent argument — that, is a preferred maximally consistent set of business rules — that supports their claim.) This phenomenon is at odds with the distaste that AI researchers usually have for credulous reasoning, but is worthy of further investigation.

3.2.4 An Open Question: The Case of Preempting Paths

Consider now — in contrast to the previous examples — inheriting wffs in a network with conflicting multiple paths. Figure 5 shows a portion of the drugs network. Although OTC Drugs (over-the-counter drugs) is a subtype of Drugs, it is not a subtype of the class of services covered by the Drugs benefit. On the other hand, a subtype of OTC Drugs, Insulin Syringes, is covered by the Drugs benefit. This is a classic case of preemption. In fact, this case is known as positive preemption, since a positive path — the direct link from Insulin Syringes to Services Covered by Drugs Benefit — preempts a negative path — the path from Insulin Syringes to OTC Drugs to Services Covered by Drugs Benefit.

Clearly, there are positive undefeated paths from Insulin Syringes to Services Covered by Drugs Benefit and from Insulin Syringes to OTC Drugs. Thus, In-

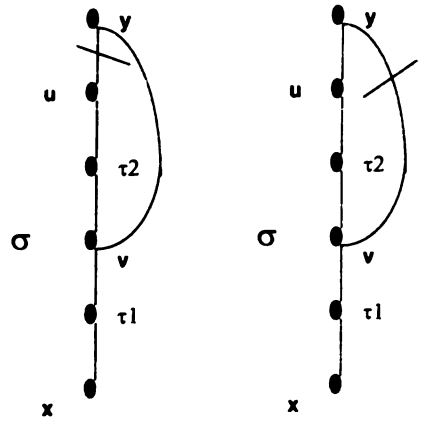


Figure 6:
a) positive path preempts negative path
b) negative path preempts positive path

sulin Syringes stands to inherit wffs from both nodes. The question is: suppose the set of wffs at OTC Drugs is inconsistent with the set of wffs at Services Covered by Drugs Benefit (either absolutely, or with respect to the background information and/or the set of wffs at Insulin Syringes). Which set of wffs do we prefer? Do we prefer the wffs from Services Covered by Drugs Benefit because that node is on the preempting path? Or do we refrain from preferring either of the paths?

There are arguments both for and against preferring nodes from the preempting path. To formally state the arguments, we use Horty's (1994) notation for preempting paths. A positive path $\pi(x, \sigma, u) \rightarrow y$ is *preempted* ((in the context (Γ, Φ)) iff there is a node v such that (i) either $v = x$ or there is a path of the form $\pi(x, \tau 1, v, \tau 2, u) \in \Phi$, and (ii) $v \not\rightarrow y \in \Gamma$. A negative path $\pi(x, \sigma, u) \not\rightarrow y$ is preempted (in the context (Γ, Φ)) iff there is a node v such that (i) either $v = x$ or there is a path of the form $\pi(x, \tau 1, v, \tau 2, u) \in \Phi$, and (ii) $v \rightarrow y \in \Gamma$. (See Figure 6.) The question is: Is the path $\pi(v, y)$ preferred to the path $\pi(v, \tau 2, u)$?

The argument for preferring a positive preempting path over the path it preempts runs as follows: Clearly, the positive path $\pi(v, y)$ is in some sense preferred to the negative path $\pi'(v, \tau 2, u, y)$ (that is the reason that we conclude y). Thus, preemption seems to offer some evidence that the preempting path is stronger than the preempted path. Then presumably, one of the links in the negative path $\pi'(v, \tau 2, u, y)$ is not as strong as the positive path between v and y. It is possible that the weak link is the negative link between u and y — but this is by no means definite, because we do, after all, conclude that u's are not y's. Thus, it seems as likely that one of the links in the path between v and u is weak. (Indeed, we know that at least in one respect v's are not typical u's — unlike typical u's, they are y's.) Thus, we ought to prefer $\pi(v, y)$ to $\pi(v, \tau 2, u)$.

On the other hand, the argument for *not* preferring a positive preempting path over the path it preempts rests on the fact that there is a positive undefeated path between v and u in the same way that there is a positive undefeated path between v and y. Thus, they both should have the same status; neither should be preferred.

If one does decide to prefer preempting over preempted paths, another question arises. Consider paths forking off from nodes in preempted paths (e.g., consider another link from u to some node z.) Ought they also to have lower priority than preempting paths? The rationale given for preferring preempting to preempted paths would seem to hold here as well, but the intuition becomes increasingly weak.

Indeed, the lack of examples makes honing intuitions particularly difficult. There are few examples in this domain of positive preemption [6] and fewer still where there are conflicting wff sets in these portions of the network; thus, it is difficult to guess what the correct behavior ought to be.

The procedure to compute the wffs that apply to a node, outlined in the previous section, takes a neutral stand on the issue. It does not state a preference for preempting paths over the paths that are preempted, but it allows that preference to be incorporated. One can incorporate such a preference in two ways: First, one can place a condition on $\mathcal{O}$ stating that preempting paths are always strictly greater than their associated preempted paths. (One may in fact wish to insist that no paths come between the preempting and preempted path.) Second, one can modify the procedure so that, when one identifies the forking paths at a fork point, one groups preempting paths with the paths they preempt, and subsequently collects wffs along preempting paths before collecting wffs along the associated preempted paths.

The difficulty is not a technical one. The hard part here is getting the intuition right. The careful examination of other domains should shed some light on the question.

3.2.5 Other Preference Criteria

The computation of $\Psi(N)$ has thus far focussed on using information present in the structure of the network. Other standard preference principles may also be needed here. One may wish to assign some wffs a higher priority than others (as in (McCarthy, 1986)), regardless of the rule's position in the network. For ex-

[6] The problem does not arise at all in cases of negative preemption — when a negative path preempts a positive path. That is because one never inherits from the last node of a negative path. In Figure 6b, one would never wish to inherit from node y — and inheriting along the path from node x to node u is simply standard wff-inheritance using the specificity criterion.

ample, it might always be the case that medical rules have higher priorities than administrative rules. Likewise, it is also reasonable to prefer a particular subset of rules based on the results that this subset entails. This is equivalent to preferring one extension, or model, over the other (as in (Shoham, 1988)). For example, we may prefer extensions in which a claim gets paid to one in which the claim does not get paid.

3.3 COMPUTATIONAL ISSUES

Inheriting rules immediately transforms the problem of inheritance from a tractable problem (at least in the case of upwards inheritance: see (Selman and Levesque, 1993)) to one that is badly intractable. Wff-inheritance in this domain will often be done in a static setting as opposed to a dynamic one, [7] and this takes some of the sting out of the fact that computing preferred maximally consistent subsets is NP-hard. Nevertheless, intractability is clearly an unpleasant issue that we must handle in some form.

In practice, we have discovered that we can deal with the complexity issue by using a divide-and-conquer strategy. The trick is to divide the set of wffs into k types, subject to the following constraint:
If Wffs(i,N1) ∪ Wffs(j,N2) is inconsistent wrt $\mathcal{B}$, then $i = j$.[8]
That is, wffs are constrained so that sets can contradict one another only within their own type. This cuts down on much consistency checking (since often when N1 $\rightsquigarrow$ N2, the wff sets at N1 are of a different type than the wff sets at N2) and greatly reduces the time needed for consistency checking and maximal subset construction and choice. Obviously, the greater k is, the more this strategy helps.

Finding a division of the wffs into sets that satisfy this constraint is not a trivial task. The wffs in this domain are taken from business rules that were originally categorized by employees of the insurance corporation into cost-share, access, administrative, and medical rules. Certain of these groups had the desired property — e.g., cost-share rules did not contradict rules of any other type — but others did not: medical rules often contradict administrative rules. On the other hand, it was possible to identify some types of administrative rules that were consistent with all rules outside of their own type. In any case, it is hardly surprising that a categorization of rules developed from a business viewpoint is not optimal in terms of efficiency.

Finding the proper knowledge representation is clearly very important. Doing so is no longer only of academic interest; it can greatly affect the tractability of the system.

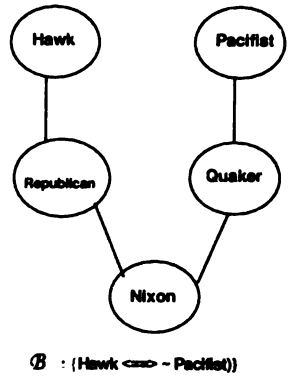


Figure 7: *The modified Nixon diamond is inconsistent, but only implicitly.*

3.4 BACK TO ATTRIBUTE INHERITANCE

We first noted the need to do conflict resolution and recognition for the purposes of wff inheritance. Further reflection, however, suggests that similar problems can arise even when performing standard attribute inheritance. Consider again Figure 1. Maternity Surgical $\rightsquigarrow$ Covered by Surgical Benefit; Maternity Surgical $\rightsquigarrow$ Covered by Maternity Benefit.

In fact, however, Maternity Surgical cannot be covered by both benefits: there is a constraint that services are covered by at most one benefit. This constraint is not explicit in the structure of the network; it is entailed by background domain knowledge; i.e., the background $\mathcal{B}$. The point of this example is not in enforcing this constraint — implementation is quite simple — but in recognizing the inconsistency. In general, when some of the knowledge of the inheritance network is present as background knowledge, inheriting attributes from multiple parents has the potential for leading to inconsistency, even if this is not explicit.

Oddly, this problem has not been discussed in the inheritance literature, perhaps because inconsistencies in the examples used have always been explicit. The Nixon diamond example (Reiter and Criscuolo, 1981) is a classic example of such an explicit inconsistency. Suppose, however, we modify Touretzky's (1986) modification of this example. (See Figure 7.) (Nixon →Republican; Nixon →Quaker; Quaker →Pacifist; Republican →Hawk). There is no explicit contradiction between Hawk and Pacifist, but if we add a statement to the background theory stating that the two concepts are contradictory, $Hawk \equiv \neg\ Pacifist$, then there is an inconsistency that must be resolved. Depending on the amount and form of the background

[7] Since the rules that apply to a medical service can be computed off-line. On the other hand, employees of the insurance corporation who modify the network may wish to see in real time how adding or changing rules at points in the network changes the rules that apply to a service.

[8] The notation Wffs(i,N) should be read: the set of wffs at node N of type i.

knowledge, detecting and resolving this inconsistency can be arbitrarily difficult (that is, as hard as the problem of wff-inheritance).

4 DISCUSSION AND RELATED WORK

4.1 COMPARING FANS TO OTHER KNOWLEDGE STRUCTURES

FANs are standard inheritance networks with wffs attached to nodes and in the background. The trend away from ontological promiscuity (the wanton creation of new knowledge structures) prompts the questions: In what way are FANs different from existing knowledge structures? Are they truly necessary or can we get by with more familiar mechanisms? This section argues that they are particularly natural for the domain at hand, and that existing structures have their own pitfalls.

FANs are extremely natural for any commercial domain in which general taxonomic information is best viewed separately from the bulk of the domain's business rules. This is precisely the situation in the medical insurance domain. The taxonomic information represents the structure of medical services by the medical profession (as well as the broad benefit categories that are standard in the insurance industry). This is a relatively static structure across products and corporations. The wffs represent the industry's business rules which change rapidly, across products and corporations. In fact, the network has proved to be especially easy for customer service representatives to master. At the same time, representing the business rules as wffs at nodes has kept the size of the network manageable (250–300 nodes).

We now examine alternatives to FANs. There are two obvious choices: representing everything in an inheritance network, or doing this entirely in a nonmonotonic logic.

Alternative I: Putting everything in the inheritance network.

As mentioned in the introduction, only some wffs can be easily represented as links between classes and attributes. However, we can capture all wffs simply by reifying each wff as a node. For example, a node $\forall x \phi(x)$ could be reified as the node representing the property of satisfying ϕ. There are several objections to this alternative.

First, many such nodes would be very unnatural, (consider the sample wffs in the first 2 sections) and quite outside the spirit of a semantic network. That is, nodes in the network, instead of representing concrete entities such as services and benefits, would represent ontologically dubious creatures.

Second, the problem of wff-inheritance has now been recast as the problem of determining if two or more paths conflict. In particular, assume wffs $\phi_1 \ldots \phi_n$ such that $\phi_1 \wedge \ldots \wedge \phi_n$ are mutually inconsistent. Let $y_1 \ldots y_n$ be the nodes associated with these wffs. It is clear that the paths $x \rightsquigarrow y_1 \ldots x \rightsquigarrow y_n$ conflict in the intuitive sense of the term. But this conflict cannot even be expressed directly in a standard Hortian or Touretzkian framework. Even if we increase expressivity, there is no obvious way to recognize and resolve this conflict. In contrast, determining maximally consistent subsets of wffs is a well-understood problem. [9]

Alternative II: Doing everything in a nonmonotonic theory That is, translate the inheritance network into a nonmonotonic theory, such as autoepistemic logic (Moore, 1985) or circumscription (McCarthy, 1980). [10]

There are several objections to this strategy. First, the distinction between static taxonomic rules and more volatile business rules has been erased. Second, this representation is much less accessible to users of the expert system, who find browsing through an inheritance network easy to learn and understand. Either they must live with this hardship or we must keep the FAN *as well as* the nonmonotonic theory. We must then either construct a mapping between the two — a non-trivial task — or be sure to always update the two structures simultaneously, an unrealistic constraint in the business world. Third, such a strategy may be computationally wasteful in that in translational theories, one must generally recompute priorities and/or specificities in a more tedious manner than the algorithms available for determining specificity in an inheritance network.

4.2 RELATED WORK

There is much work in the literature on determining maximally consistent sets of defaults (Brewka, 1989, Geffner, 1990, Grosof, 1991, Delgrande and Schaub, 1994, among others). Such work is obviously very close in spirit to ours. In particular, there is an emphasis in

[9] As mentioned in the previous section, the general problem of determining if paths conflict already exists in many networks, such as the modified Nixon diamond of Figure 7. While we see that these problems can beset any network, we nevertheless believe that flooding networks with this problem, which would happen if wffs were reified as nodes, is not desirable. This is particularly true since the problem of determining if arbitrary paths in a network are inconsistent with respect to one another has not been addressed.

[10] There are a variety of methods for doing this such as (Gelfond, 1990). However, see also Horty (1994), who argues that specifying a translational semantics for a path-based inheritance network is a problem that is not yet completely solved. In any case, it is certainly possible, given a *particular* network, to give an equivalent nonmonotonic theory.

these works on considerations of specificity. This paper differs from these works in a number of respects (such as the consideration of non-conflicting-path multiple inheritance). Most importantly, what distinguishes this paper from these works is the emphasis on the semantic network structure which we take as our starting point. The other works take some nonmonotonic system as their starting point. The difference is analogous to the distinction that Horty (1994) points out between path-based and translational theories of inheritance; the latter specify the meaning of a network in terms of a nonmonotonic formalism; the former specify the meaning in terms of the paths themselves. Just as Horty argued for the naturalness and intuitiveness of path-based approaches for standard inheritance, we argue for the ease, naturalness, and intuitiveness of the path-based approach to wff-inheritance. In particular, note the discussion in (Horty, 1994), which argues that researchers using the translational approaches have not yet satisfactorily formalized the concept of specificity, which is easy to formalize within a path-based approach.

We do not, however, believe that these systems should be seen as competing. Rather, we believe that the following is the case: A major task in knowledge representation is the articulation of the structure that is most natural and useful for a particular application. Another major task is demonstrating, if possible, that this structure is equivalent to more familiar structures, or if that is not possible, demonstrating the inequivalence. In this spirit, we suggest that formula-augmented semantic networks have a useful role to play in domains in which taxonomic knowledge is a major, but not sole, component of the domain knowledge. (We are currently using FANs in life insurance and property & casualty applications). At the same time, we hope that future research will shed light on the equivalences between FANs and the systems of Geffner, Grosof, and others.

Acknowledgements: Thanks to Ernie Davis, David Etherington, Benjamin Grosof, Daniel Lehmann, Moninder Singh, and Rich Thomason for helpful discussions and suggestions, and to Bill Wynne for his assistance in populating the benefits semantic network.

REFERENCES

G. Brewka (1989): Preferred Subtheories: An Extended Logical Framework for Default Reasoning, *IJCAI 1989*, 1043-1048

A. Darwiche and J. Pearl (1994): On the Logic of Iterated Belief Revision, in R. Fagin (ed): *Proceedings of the Fifth Conference of Theoretical Aspects of Reasoning About Knowledge*, Morgan Kaufmann, San Mateo, 5-23

J. Delgrande and T. Schaub: A General Approach to Specificity in Default Reasoning, *KR 94*, 146-157

D. Etherington(1988): *Reasoning with Incomplete Information*, Morgan Kaufmann, Los Altos

P. Gardenfors (1988): *Knowledge in Flux*, MIT Press, Cambridge

H. Geffner (1990): *Default Reasoning: Casual And Conditional Theories*, MIT Press, Cambridge

M. Gelfond and H. Przymusinska (1990): Formalization of Inheritance Reasoning in Autoepistemic Logic, *Fundamenta Informaticae 13*, 1990, 403-443

B. Grosof (1991): Generalizing Prioritization, in J. Allen, R. Fikes, and E. Sandewall (eds): *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, San Mateo, 289-300

J. Horty (1994): Some Direct Theories of Nonmonotonic Inheritance in D. Gabbay, C. Hogger, and J. Robinson, eds: *Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 3: Nonmonotonic Reasoning and Uncertain Reasoning*, Oxford University Press, Oxford, 111-187

J. Horty, R. Thomason and D. Touretzky (1990): A Skeptical Theory of Inheritance in Nonmonotonic Semantic Networks, *Artificial Intelligence 42*, 311-349

H. Kautz and B. Selman (1989): Hard Problems for Simple Default Logics in R. Brachman, H. Levesque, and R. Reiter (eds): *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufman, San Mateo, 189-197

J. McCarthy (1980): Circumscription – A Form of Non-Monotonic Reasoning, *Artificial Intelligence 13*, 27-39

J. McCarthy (1986): Applications of Circumscription to Formalizing Common-sense Knowledge, *Artificial Intelligence 28*, 86-116

R. Moore (1985): Semantical Considerations on Nonmonotonic Logic, *Artificial Intelligence*, 25(1), 75-94

R. Reiter and G. Criscuolo (1981): On Interacting Defaults, *IJCAI 1981*, 270-276

B. Selman and H. Levesque (1993): The Complexity of Path-Based Defeasible Inheritance, *Artificial Intelligence 62:2*, 303-340

Y. Shoham (1988): *Reasoning About Change: Time and Causation from the Standpoint of Artificial Intelligence*, MIT Press, Cambridge

L. Stein (1992): Resolving Ambiguity in Nonmonotonic Inheritance Hierarchies, *Artificial Intelligence 55*, 259-310

D. Touretzky (1986): *The Mathematics of Inheritance Systems*, Morgan Kaufmann, Los Altos

Partial Orders of Sorts and Inheritances

(or Placing Inheritance in Context)

Nirad Sharma
Department of Computer Science
The University of Queensland
Brisbane QLD 4072. Australia
niradcs.uq.edu.au

Abstract

We consider the role of formalised contexts for clarifying subsumption and inheritance in concept taxonomies. Inheritance is traditionally thought of as following a single rooted directed acyclic graph formed by partial orders based on concept subsumption relationships. We dispute this as a simplification, noting that while the subsumption ordering between two concepts may be based on certain properties modelled for the concepts relative to some context, the orderings that induce the paths down which inheritance of properties is desired may not always be based on the same properties. The kinds of problems this leads to are often witnessed when modelling large or rich concept taxonomies - context-naive inheritance starts to behave restrictively and unintuitively, requiring artificial concepts to be introduced into the taxonomy that obscure or even lose some of the true subsumption relationships.

This paper investigates how the introduction of contexts as formal identifiers and then stating theories relative to contexts can provide a more flexible mechanism for managing inheritance and ontologies in the framework of an order-sorted first order logic.

keywords : context, order-sorting, ontology. modality, knowledge sharing

1 Introduction

When developing theories (well-formed formulae in some logic) of artifacts and events in the world, one does so with a particular application in mind - ontologies are not usefully modelled independent of usage. To do so requires statement of every possible qualification for the concepts of the model, a clearly infeasible task. As such, concepts are always modelled at some level of abstraction and perspective as relevant to their application's requirements.

The observation made by McCarthy (McCarthy, 1990) and, more recently, in (McCarthy, 1993; Nayak, 1994; Giunchiglia and Traverso, 1995; Attardi and Simi, 1995; Sharma, 1995) was that this leads to problem-solvers being brittle and inflexible, lacking "generality". Simply, most problem-solvers are constructed with a single domain theory that necessarily encodes the assumptions of the task for which it was axiomatised, making certain representational commitments. For example, a theory for local area ethernet network configuration may not be general enough to deal with networks having both AUI and BNC cabling. These implicit assumptions are not only inevitable given the impracticality of modelling a real-world domain in complete generality but desirable as this very simplification engenders representation and reasoning efficiencies.

Often, a problem-solver will require concepts to be organised into several taxonomies from different perspectives, as is commonly required in configuration problems (Mittal and Frayman, 1987). Rather than attempt construction of problem-solvers that can cope with all situations in complete generality, an impossible task, a more cogent approach would be to populate a problem-solver with several theories of a problem domain (Sharma, 1995). By explicating the context in which a theory is stated, one is then able to represent multiple theories, relate their formulae and reason between the theories in a uniform formal framework.

In this paper, context-naive treatments of subsumption and inheritance are examined. When one considers that concept subsumption depends on the concepts' definitional properties and constraints (explicitly stated or implicit) and that these properties and constraints are themselves context-dependent, some insights into the problems of [context-naive] subsumption and inheritance emerge. An approach for clarifying subsumption taxonomies is proposed with formulae (concepts, subsumption, definitions) being stated rel-

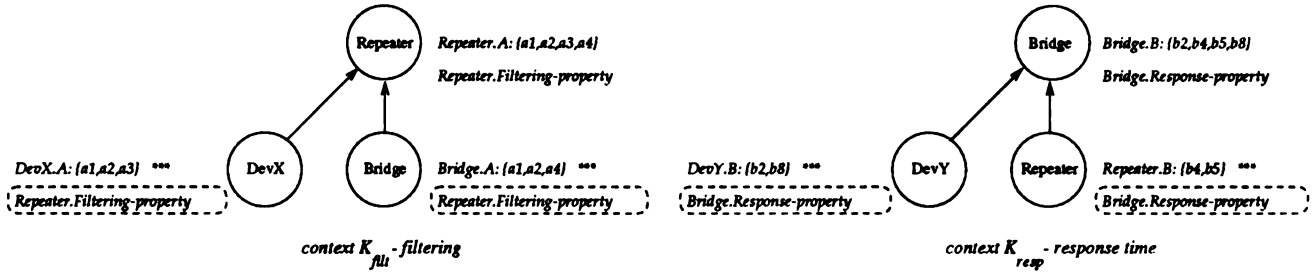


Figure 1: **Repeater** and **Bridge** with additional context-specific concepts and properties

ative to contexts in the framework of an order-sorted first order logic. As the logic is modal in nature, extensibility to other modalities is facilitated. Traditionally overlooked, these issues are fundamental to the construction of practically useful and reusable ontologies for multi-function knowledge bases.

1.1 A simple example of context-naive inheritance

We consider a fragment of a more extensive theory of internetworking from (Sharma, 1996). A **Repeater** indiscriminantly forwards packets appearing on the ethernet-segment of one of its interfaces to all other ethernet-segments to which it has interfaces. A **Bridge** will look at the destination ethernet address of the packet and, if it knows that the address is reachable at one of its attached networks, forwards it there. Otherwise, it acts like a repeater and sends the packet to all attached networks. A bridge can be seen as a specialised repeater in that it provides additional filtering facilities or, alternatively, a repeater may be seen as a specialised bridge in that the per-packet response time for a repeater may range over a smaller subrange of that for a bridge. Such substitutability decisions are prevalent for configuration tasks (Sharma, 1996; Mittal and Frayman, 1987).

Consider (Figure 1) that in the context of *filtering*, the devices **Bridge** and **DevX** specialise **Repeater** w.r.t. property **A** and thus inherit **Filtering-property** from **Repeater**. Specialisation is illustrated by constraining the sample extensions of **A** e.g. $\{a1, a2, a3\} \subset \{a1, a2, a3, a4\}$, the constrained attribute(s) marked '***'. In the context of *response times*, the devices **Repeater** and **DevY** specialise **Bridge** w.r.t. property **B** and thus inherit **Response-property**.

It is not clear which of **Repeater** and **Bridge** would be a subtype of the other if modelled in a flat concept space. That is, one type is a subtype of (or specialises) another *relative to some properties in a context*. While the conventional approach would be to introduce two artificial concepts **NetLink1** and **NetLink2** (Figure 2), the subsumption relationships that existed between **Bridge** and **Repeater** in the *filtering* and *response time* contexts would be lost, particularly if they had been primitive subconcepts (cf. KL-ONE (Woods and Schmolze, 1992)). Such phenomena are problematic for the construction of practically reusable ontologies expected to have more general application and for large, multi-perspective knowledge bases.

At a particular level of abstraction, certain concepts and properties (essentially, relations with instances of the concept) will be *relevant*. Necessary and sufficiency conditions for a concept or relation are thus similarly due to a particular perspective or level of abstraction. By convention, all property inheritance for a concept is considered to follow from a single partial order for the subsumption relationships, a treatment oversimplified in practice. This causality can be characterised as follows :

1. relevant abstraction level and perspective
   ⇓ *induces*
2. qualifications/properties to be modelled
   ⇓ *induces*
3. subsumption orderings over sorts
   ⋮
4. inheritance orderings

While it is always possible to create a more general artificial concept for two concepts (Figure 2) given the dilemma of Figure 1, more problems arise:

1. Constraints need to be placed in sub-concepts of the artificial generalised concept if it is necessary to restrict the availability of the inherited properties to their specific contexts (i.e. placing *if active-context* K_{filt} on properties in Figure 2). A modelling mismatch occurs, particularly for object-oriented methodologies which typically prescribe method definitions be hidden behind method signatures. The signature misrepresents a concept's properties, hindering clarity e.g. {**A**,**B**,**Filtering-property**,**Response-property**} for **Bridge** and **Repeater**.
2. The perspectives (K_{filt}, K_{resp}) relative to which the orderings between **Repeater** and **Bridge** were captured may be relevant to other type orderings or property inheritances e.g. for **Media-Types** or

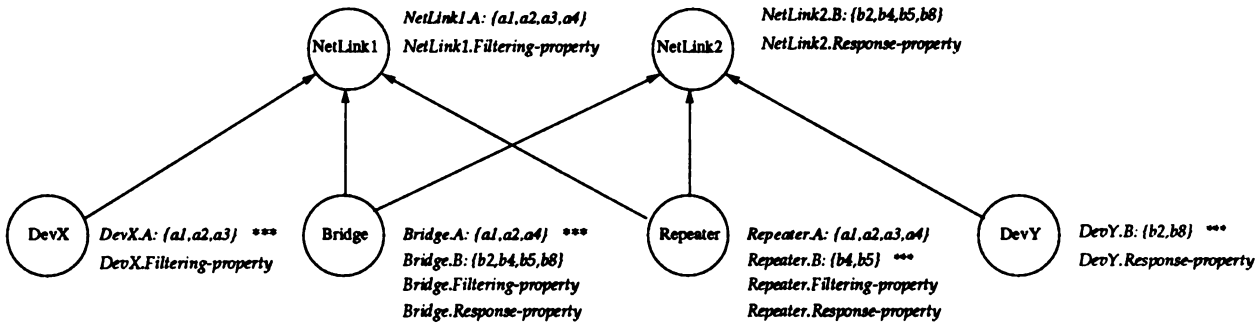


Figure 2: **Repeater** and **Bridge** with concepts in a flattened concept space

Connectors. A reasoner working from a response-time perspective (K_{resp}) may want to reason with the concepts and properties relevant to the K_{resp} perspective.

Ideally, the ontology specification language should support multiple relevant abstraction levels and perspectives and thus the partial orders of sorts that are appropriate. We examine the problem in more detail and then present a generalised solution in the form of an order-sorted logic extended with context identifiers.

2 Contexts

The common thread to these problems is that concept subsumption partial orders are *relevant within contexts* and that these contexts have relationships to each other. The introduction of contexts as formal objects into a representation scheme makes it possible to deal with different theories (sets of *wffs*) and their languages (order-sorted predicate and function symbols) being captured relative to different sets of background assumptions (perspectives, levels of detail) and to represent and reason with and between these theories in one formal system (Buvač et al., 1995; Sharma, 1995). A multi-dimensional aspect to concept taxonomies can then be introduced.

We define a countably infinite set $\mathcal{K}$ of *context symbols*, denoted $\kappa_0..\kappa_\omega$. A context scoping defines a language (signature) and set of models that characterise the valid states of affairs for the formulae of that context. All formulae are stated relative to some context scoping. For example, to state that a formula ϕ holds in the context κ_{filt}, we write $\mathsf{in}(\kappa_{filt}, \phi)$.

Context-embedded formulae are themselves *wffs*, allowing arbitrary deepness of embedding. It is not enough to just consider some symbol's meaning as being relative to a context but to the path by which that context was reached (the context of the context). For example, the symbols and their meanings in the networks context (κ_{netw}) from the perspective of 1985 (κ_{1985}) may be quite different to the symbols in κ_{netw} in 1995 (κ_{1995}).

This immediately raises the issue of some outermost context or "absolute truth" relative to which all *wffs* are ultimately embedded. Rather than enter the domain of philosophers and linguistics with what is a variant of the ubiquitous symbol grounding problem in artificial intelligence, we fix an outermost context $\kappa_\top$. Otherwise, we find ourselves grappling with the "infinite regress" (Quine, 1969) of trying to fix the base language for a sequence of languages, each providing the background assumptions and denotation for its predecessor in the sequence. Our assumption of an implicit outermost context is analogous to the fact that designers of engineering artifacts typically don't trouble themselves with the situatedness of their systems. Interestingly, the treatments of embedded and order-sorted contexts here seem to correspond to Quine's ideas of "subordinate theories" whose languages are relative to those of base theories.

We define $\mathcal{K}^*$ as the set of all context sequences formed from $\mathcal{K}$. A *context identifier* $\overline{\kappa}$ (borrowing from (Buvač et al., 1995)) is a sequence $[\kappa_1, ..., \kappa_n]$ that ranges over $\mathcal{K}^*$, denoting a successive embedding in contexts κ_1 to κ_n. The singleton sequence $[\kappa_i]$ may be abbreviated κ_i while the notation $\overline{\kappa_i} * \overline{\kappa_j}$ denotes sequence concatenation.

The formula $\mathsf{in}(\overline{\kappa}, \phi)$ where $\overline{\kappa}$ is some arbitrary context sequence and ϕ is a *wff* can thus be rewritten :

$$\begin{array}{lcll} \mathsf{in}([], \phi) & := & \phi & \\ \mathsf{in}([\kappa_i], \phi) & := & \mathsf{in}(\kappa_i, \phi) & \\ \mathsf{in}([\kappa_i, ..., \kappa_j], \phi) & := & \mathsf{in}(\kappa_i, \mathsf{in}([\kappa_{i+1}, ..., \kappa_j], \phi)) & \\ & & & i < j \leq n \end{array}$$

The inductive style of this description alludes to the iterative scoping provided by each successive in embedding. Thus, $\mathsf{in}(\kappa_{1985}, \mathsf{in}(\kappa_{netw}, \phi))$ can be rewritten $\mathsf{in}([\kappa_{1985}, \kappa_{netw}], \phi)$.

Clearly, it is not satisfactory to be able to state that any arbitrary context symbol can hold in an arbitrary context scoping e.g. κ_{netw} being nonsensical in κ_{1746}. Rather, the allowable relationships between contexts identifiers will be captured by context relationships, populating a *context space* of allowable context transitions. (Context spaces, relationships and their roles in

directing reasoning will be presented in a subsequent paper. This work deals with the formal foundations of a context-augmented language rather than its population.)

A further consequence of in forming a *wff* is that the notion of context extends that of modularity as found in conventional programming languages. Term designators are rigid and thus object variables can be quantified outside of and bind variables in context embedded formulae, enabling formulae to be related across contexts. For example,

$$\forall x : \mathit{Bridge} \cdot \mathsf{in}(\kappa_{filt}, \mathit{Prop}_1(x)) \equiv \mathsf{in}(\kappa_{resp}, \exists s : \mathit{Sit} \cdot \mathit{Prop}_1(x, s))$$

states that for all objects x in the semantic domain of the context the formula is declared in, $\mathit{Prop}_1(x)$ is true in context κ_{filt} iff $\exists s : \mathit{Sit} \cdot \mathit{Prop}_1(x)$ is true in context κ_{resp}. As the quantification $\forall x$ is outside the scopes of the embedded contexts, the objects that x can bind to must be interpreted sensibly within the embedded contexts. For convenience, we will assume constant domains of interpretation for all context sequences (§3.2).

While such *lifting rules* are convenient for relating individual formulae, individuating a lifting rule for each symbol or, more generally, formula in large contexts would become overly cumbersome. To provide a convenient mechanism for incremental signature and theory construction, inheritance between contexts is required. Order-sorting just the context symbols is too restrictive, however, as this restricts context inheritance to within the scope of the enclosing context sequences. Rather, the logic should enable a context symbol to be "prototyped" from, potentially, outside a context symbol's scoping context sequence.

As an example, we may wish for all context sequences terminated by context symbol κ_{netw} to include a prototypical signature and formulae for network contexts irrespective of context scoping e.g. in $[\kappa_{1986}, \kappa_{netw}]$ and $[\kappa_{1996}, \kappa_{netw}]$. Thus, we generalise context order-sorting to a partial order $\ll^{\kappa}$ over context sequences ($\ll^{\kappa} \subseteq \mathcal{K}^* \times \mathcal{K}^*$). This will become particularly important for flexible construction of context spaces. Clearly, participation in $\ll^{\kappa}$ will have consequences for the respective context sequences' signatures and models.

3 An order-sorted logic for context spaces

The design issues discussed so far are now captured in an order-sorted first order logic upon which context spaces can be built for flexible ontology construction. While an order-sorted first order logic is a conservative extension of a non-sorted first order logic and concepts such as in Figure 1 can be represented as unary predicates, the virtues of representing explicit sorts and subsorts are well-understood. Order-sorting formulae facilitates clearer and more concise specification of ontologies by explicating taxonomic structure, simplifying the expression of formulae, and enabling specialised reasoning procedures to exploit the sortal structure to more efficiently direct reasoning, simplifying proofs.

The logic supports context sequences and order-sorted context sequences with all signatures and sentences defined relative to context sequences. Treatment of terms as rigid designators facilitates cross-context identification for lifting rules. A Hilbert-style system is defined to simplify presentation and some notational conventions borrowed from Buvač (Buvač et al., 1995) and Beierle (Beierle et al., 1992).

3.1 Syntax

We define two distinct countably infinite sets $\mathcal{K}$ and $\mathcal{L}$ that are the set of context symbols and the set of all predicate, function and constant symbols, respectively. $\mathcal{K}^*$ is the set of all sequences of context symbols of $\mathcal{K}$. A context identifier denoted $\overline{\kappa} \in \mathcal{K}^*$ identifies a context scoping relative to which signatures and formulae are stated (§2).

If we consider K to be the set of all valid context sequences ($K \subset \mathcal{K}^*$), Σ^K denotes the set of all context signatures in the context space where $\Sigma_{\overline{\kappa}}$ denotes the signature for a context identified by $\overline{\kappa}$:

$$\Sigma^K = \{\Sigma_{\overline{\kappa}} \mid \overline{\kappa} \in K\}$$

The outermost context is a distinguished context sequence $[\kappa_\top]$, abbreviated to $\overline{\kappa_\top}$. The significance of $\overline{\kappa_\top}$ is given that terms and thus constants are rigid designators, some context outer to all other contexts must exist to define the symbols and sorts of the constants. As such, an axiom of the system is that all contexts $\overline{\kappa} \in K$ inherit from $\overline{\kappa_\top}$.

In principle, one could consider the signature identified by the empty context sequence [] ($\Sigma_{[]}$) as providing the prototype signature for all other contexts. We consider this approach to be unsound as it implicitly posits the existence of an absolute truth rather than recognising the situatedness of any formal ontology (§2). Furthermore, an axiom stating that all contexts inherits from the context [] would still be required.

A valid context identifier $\overline{\kappa}$ has the signature $\Sigma_{\overline{\kappa}} = \langle\langle \mathcal{S}_{\overline{\kappa}}, \ll_{\overline{\kappa}}\rangle, \mathcal{P}_{\overline{\kappa}}, \mathcal{F}_{\overline{\kappa}}, \mathcal{C}_{\overline{\kappa}}\rangle$ where each of $\mathcal{S}_{\overline{\kappa}}$, $\mathcal{P}_{\overline{\kappa}}$, $\mathcal{F}_{\overline{\kappa}}$ and $\mathcal{C}_{\overline{\kappa}}$ are drawn from $\mathcal{L}$. We assume for convenience that for a context identifier $\overline{\kappa}$, the sets $\mathcal{S}_{\overline{\kappa}}$, $\mathcal{P}_{\overline{\kappa}}$, $\mathcal{F}_{\overline{\kappa}}$ and $\mathcal{C}_{\overline{\kappa}}$ are pairwise disjoint, defined as :

- $\langle \mathcal{S}_{\overline{\kappa}}, \ll_{\overline{\kappa}}\rangle$: The set of sorts $\mathcal{S}_{\overline{\kappa}}$ in $\overline{\kappa}$ and their set of subsumption partial orders $\ll_{\overline{\kappa}}$ in $\overline{\kappa}$ ($\ll_{\overline{\kappa}} \subseteq \mathcal{S}_{\overline{\kappa}} \times \mathcal{S}_{\overline{\kappa}}$) that forms the sort lattice for context

$\overline{\kappa}$. The least and greatest elements of the lattice are $\bot$ and $\top$. $\mathcal{S}^*_{\overline{\kappa}}$ will denote the set of all sequences over $\mathcal{S}_{\overline{\kappa}}$. Cycles in the sort taxonomy are forbidden for $\ll_{\overline{\kappa}}$.

- $\mathcal{P}_{\overline{\kappa}} = \{\mathcal{P}^a_{\overline{\kappa}} \mid a \in \mathcal{S}^*_{\overline{\kappa}}\}$: An $\mathcal{S}^*_{\overline{\kappa}}$-indexed family of sets of predicate symbols where $p \in \mathcal{P}^a_{\overline{\kappa}}$ has the signature $p : S_1 \times ... \times S_n$ with $a = S_1, ..., S_n$.
- $\mathcal{F}_{\overline{\kappa}} = \{\mathcal{F}^{a,S}_{\overline{\kappa}} \mid a \in \mathcal{S}^*_{\overline{\kappa}}, S \in \mathcal{S}_{\overline{\kappa}}\}$: An $(\mathcal{S}^*_{\overline{\kappa}} \times \mathcal{S})$-indexed family of sets of function symbols where $f \in \mathcal{F}^{a,S}_{\overline{\kappa}}$ has the signature $f : S_1 \times ... \times S_n \rightarrow S$ with $a = S_1, ..., S_n$.
- $\mathcal{C}_{\overline{\kappa}} = \{\mathcal{C}^S_{\overline{\kappa}} \mid S \in \mathcal{S}_{\overline{\kappa}}\}$: An $\mathcal{S}_{\overline{\kappa}}$-indexed set of constant symbols where $c \in \mathcal{C}^S_{\overline{\kappa}}$ has the signature $c : S$. $\mathcal{C}_{\overline{\kappa}}$ differs from $\mathcal{F}^{\epsilon,}_{\overline{\kappa}}$ (ε denotes an empty sequence of sort symbols) as constants are rigid designators while functions are non-rigid (§3.2). It is not meaningful to extend $\mathcal{C}_{\overline{\kappa}}$ beyond $\mathcal{C}_{\overline{\kappa\top}}$, constants being rigid designators, thus $\forall \overline{\kappa_1}, \overline{\kappa_2} \in K \cdot \mathcal{C}_{\overline{\kappa_1}} = \mathcal{C}_{\overline{\kappa_2}}$.

For all $\overline{\kappa} \in K$, the equality symbol $\doteq$ is a binary relation whose arguments are of sort $\top$ i.e. $\doteq \in \mathcal{P}^{\top,\top}_{\overline{\kappa}}$.

Context identifiers can be order-sorted ($\overline{\kappa_1} \ll^\kappa \overline{\kappa_2}$) to state that $\overline{\kappa_2}$ inherits from and thus may extend the signature and set of formulae from $\overline{\kappa_1}$. If $\overline{\kappa_1} \ll^\kappa \overline{\kappa_2}$ ($\overline{\kappa_2}$ specialises $\overline{\kappa_1}$) then $\mathcal{S}_{\overline{\kappa_1}} \subseteq \mathcal{S}_{\overline{\kappa_2}}$, $\mathcal{P}_{\overline{\kappa_1}} \subseteq \mathcal{P}_{\overline{\kappa_2}}$, $\mathcal{F}_{\overline{\kappa_1}} \subseteq \mathcal{F}_{\overline{\kappa_2}}$ and $\mathcal{C}_{\overline{\kappa_1}} \subseteq \mathcal{C}_{\overline{\kappa_2}}$ (more precisely, $\mathcal{C}_{\overline{\kappa_1}} = \mathcal{C}_{\overline{\kappa_2}}$). That is, where one context identifier specialises another, it may introduce new sort, predicate or function symbols (as well as introducing new sentences). $\ll^\kappa$ is reflexive, transitive and anti-symmetric.

The resultant context space Σ^κ is

$$\Sigma^\kappa = \langle\langle K, \ll^\kappa\rangle, \Sigma^K\rangle$$

Recall that all context sequences inherit the signature of $\overline{\kappa\top}$ and, as the outermost context, $\overline{\kappa\top}$ inherits from no other context sequence i.e. $\forall \overline{\kappa} \in K \cdot (\overline{\kappa\top}, \overline{\kappa}) \in \ll^\kappa$ and $\nexists \overline{\kappa} \in K \cdot (\overline{\kappa}, \overline{\kappa\top}) \in \ll^\kappa$.

Note that a symbol may be an element of $\mathcal{P}_{\overline{\kappa_1}}$ and $\mathcal{F}_{\overline{\kappa_2}}$ where $\overline{\kappa_1} \neq \overline{\kappa_2}$, assuming $\overline{\kappa_1}$ and $\overline{\kappa_2}$ are not bound by $\ll^\kappa$, as a symbol's meaning may differ in different contexts. Similarly, a symbol's arity may vary between non-bound contexts. For example, the symbol *enabled* may be a unary predicate denoting a device status in κ_{filt} but may be a binary predicate denoting an action in some other context.

It is important to note that for two arbitrary context sequences $\overline{\kappa_i}, \overline{\kappa_j} \in K$ where $\overline{\kappa_j}$ extends $\overline{\kappa_i}$ ($\overline{\kappa_j} = \overline{\kappa_i} * \overline{\kappa_\Delta}$ for some non-empty $\overline{\kappa_\Delta}$) and is thus embedded in $\overline{\kappa_i}$, that is where $\mathsf{in}(\overline{\kappa_j}, \phi) \equiv \mathsf{in}(\overline{\kappa_i}, \mathsf{in}(\overline{\kappa_\Delta}, \phi))$ for an arbitrary ϕ, no relationship necessarily exists between $\Sigma_{\overline{\kappa_i}}$ and $\Sigma_{\overline{\kappa_j}}$ as long as $\overline{\kappa_i}$ and $\overline{\kappa_j}$ are not related by $\ll_\kappa$.

Variables and terms are defined independent of context identifiers as rigid designators to enable cross-context references. A problem is that variables exist independent of the signatures of the contexts in which they are bound, making the definition of well-sortedness of terms problematic. Our solution is to stay consistent with the earlier assertion that a context space and thus all its contexts and formulae are situated relative to some outermost context $\overline{\kappa\top}$. Variables are considered to be sorted relative to $\Sigma_{\overline{\kappa\top}}$.

V_S denotes the set of all variables of sort S in context $\overline{\kappa\top}$ ($S \in \mathcal{S}_{\overline{\kappa\top}}$ from $\Sigma_{\overline{\kappa\top}}$). By virtue of $\forall \overline{\kappa} \in K \cdot (\overline{\kappa\top}, \overline{\kappa}) \in \ll^\kappa$, the sortal structure of $\overline{\kappa}$ will be present in all context sequences of K, in effect providing a "base" type system and facilitating appropriate treatment of well-sortedness for variables. The class of variables $V_{\overline{\kappa\top}}$ (abbreviated to V) for $\overline{\kappa\top}$ is $V = \{V_S \mid S \in \mathcal{S}_{\overline{\kappa\top}}\}$.

Well-sorted terms can be defined on a context-specific basis to account for context signatures. The set $\mathbb{T}_{\Sigma_{\overline{\kappa}}}(V)$ of well-sorted terms for a context $\overline{\kappa}$ is the smallest S-indexed family of sets constructed as follows (where $S \in \mathcal{S}_{\overline{\kappa}}$) :

- $v \in \mathbb{T}_{\Sigma_{\overline{\kappa}}}(V)_S$ for each $v \in V_S$
- $c \in \mathbb{T}_{\Sigma_{\overline{\kappa}}}(V)_S$ for each $c \in \mathcal{C}^S$
- $c \in \mathbb{T}_{\Sigma_{\overline{\kappa}}}(V)_S$ for each $c \in \mathcal{F}^{\epsilon,S}_{\overline{\kappa}}$. c ranges over the 0-ary functions of context $\overline{\kappa}$, providing context-specific constants
- $f(t_1, ..., t_n) \in \mathbb{T}_{\Sigma_{\overline{\kappa}}}(V)_S$ where $t_1, ..., t_n$ are terms in $\mathbb{T}_{\Sigma_{\overline{\kappa}}}(V)$ of sort $R_1, ..., R_n$, respectively, if $f \in \mathcal{F}^{S_1...S_n,S}_{\overline{\kappa}}$ and $\forall i : 1..n \cdot R_i \ll_{\overline{\kappa}} S_i$

A term not containing any variables is ground, as is a formula all of whose terms are ground.

The family of *well-formed formulae* $\mathbb{F}$ is the least set defined by :

$$\mathbb{F}_0 \cup (\neg \mathbb{F}) \cup (\mathbb{F} \supset \mathbb{F}) \cup (\forall \mathcal{V} : \mathcal{S} \cdot \mathbb{F}) \cup (\mathsf{in}(\mathcal{K}, \mathbb{F}))$$

where $\mathbb{F}_0$ are the set of atomic formulae (i.e. predicates symbols with arguments). The usual abbreviations hold for $\wedge$, $\vee$, $\equiv$ and $\exists$.

The family of *well-sorted formulae* $\mathbb{F}_{\Sigma^K}$ ($\subseteq \mathbb{F}$) are defined for the collection Σ^K of context signatures in a context space. We first define $\mathbb{F}_{\Sigma_{\overline{\kappa}}}$, the set of well-sorted formulae for the context identified by $\overline{\kappa}$, inductively over the structure of formulae stated in $\overline{\kappa}$ as the minimal set s.t. :

- $p(t_1, ..., t_n) \in \mathbb{F}_{\Sigma_{\overline{\kappa}}}$ where $t_1, ..., t_n$ are terms in $\mathbb{T}_{\Sigma_{\overline{\kappa}}}(V)$ of sort $R_1, ..., R_n$, respectively, if $p \in \mathcal{P}^{S_1...S_n}_{\overline{\kappa}}$ and $\forall j : 1..n \cdot R_j \ll_{\overline{\kappa}} S_j$
- $\neg\phi \in \mathbb{F}_{\Sigma_{\overline{\kappa}}}$ if $\phi \in \mathbb{F}_{\Sigma_{\overline{\kappa}}}$

- $\phi \supset \psi \in \mathbb{F}_{\Sigma_{\overline{\kappa}}}$ if $\phi \in \mathbb{F}_{\Sigma_{\overline{\kappa}}}$ and $\psi \in \mathbb{F}_{\Sigma_{\overline{\kappa}}}$
- $\mathsf{in}(\kappa_i, \phi) \in \mathbb{F}_{\Sigma_{\overline{\kappa}}}$ if $\kappa_i \in K$ and $\phi \in \mathbb{F}_{\Sigma_{\overline{\kappa} * \kappa_i}}$
- $\forall v : S \cdot \phi \in \mathbb{F}_{\Sigma_{\overline{\kappa}}}$ if $v \in V_S$, $\phi \in \mathbb{F}_{\Sigma_{\overline{\kappa}}}$ and for all occurrences of v bound in ϕ filling an argument of sort R, $S \ll_{\overline{\kappa}} R$.

Variables appearing in a formula not within the scope of any quantifier are free while a formula not containing any free variables is a *sentence*.

The set of $\mathbb{F}_{\Sigma_{\overline{\kappa}}}$ for all $\overline{\kappa} \in K$ in a context space trivially partitions $\mathbb{F}_{\Sigma K}$ which can thus be reconstructed as $\mathbb{F}_{\Sigma K} = \bigcup_{\overline{\kappa} \in K} \mathbb{F}_{\Sigma_{\overline{\kappa}}}$.

3.2 Semantics

Our semantics are set-theoretic and closely follow the classical semantics of conventional order-sorted first order systems. A context identifier identifies a set of models that characterise the states of affairs captured by the context. While it is desirable that different context sequences be interpreted over differing domains, in this logic the domain of interpretation $\mathcal{D}$ is fixed for all contexts.

Recall (§3.1) that a context identifier $\overline{\kappa}$ identifies a signature $\Sigma_{\overline{\kappa}} = \langle \langle \mathcal{S}_{\overline{\kappa}}, \ll_{\overline{\kappa}} \rangle, \mathcal{P}_{\overline{\kappa}}, \mathcal{F}_{\overline{\kappa}}, \mathcal{C}_{\overline{\kappa}} \rangle$. An interpretation $\nu_{\overline{\kappa}}$ for context identifier $\overline{\kappa}$ is defined as a structure containing :

- an $\mathcal{S}_{\overline{\kappa}}$-indexed set $\{C_S \mid S \in \mathcal{S}_{\overline{\kappa}}\}$ where C_S is the *carrier set* ranging over $\mathcal{D}$ for interpreting the sort S. Note that for all context identifiers $\overline{\kappa}$, $C_\perp = \varnothing$, $C_\top = \mathcal{D}$ and where $R, S \in \mathcal{S}_{\overline{\kappa}}$ and $R \ll_{\overline{\kappa}} S$, $C_R \subseteq C_S$.
- an $\mathcal{S}_{\overline{\kappa}}{}^*$-indexed family of relations $\{p_\nu \subseteq C_{S_1} \times \ldots \times C_{S_n} \mid p : S_1 \ldots S_n \in \mathcal{P}_{\overline{\kappa}}\}$
- an $\mathcal{S}_{\overline{\kappa}}{}^* \times \mathcal{S}_{\overline{\kappa}}$-indexed family of functions $\{f_\nu : C_{S_1} \times \ldots \times C_{S_n} \to C_S \mid f : S_1 \ldots S_n \to S \in \mathcal{F}_{\overline{\kappa}}\}$
- an $\mathcal{S}_{\overline{\kappa_\top}}$-index family of 0-ary functions for constants $\{c_\nu :\to C_S \mid c : S \in \mathcal{C}_{\overline{\kappa_\top}}\}$
- the equality symbol $\doteq$ is a distinguished binary relation symbol defined for all contexts interpreted as $\doteq = \{(o, o) \mid o \in C_\top\}$

The family of all interpretations for a context $\overline{\kappa}$ is denoted $\mathfrak{I}(\overline{\kappa})$, that is $\nu_{\overline{\kappa}} \in \mathfrak{I}(\overline{\kappa})$. $\mathfrak{M}(\overline{\kappa})$ $(\subseteq \mathfrak{I}(\overline{\kappa}))$ denotes the family of interpretations that are consistent with and are thus models for the formulae of context $\overline{\kappa}$. Note that because constants and terms are rigid designators, all interpretations for all contexts will evaluate a constant symbol to the same domain element.

We can now introduce the entailment relation $\models_{\overline{\kappa}}$ over the interpretations and well-sorted formulae for a context sequence $\overline{\kappa}$ in a context space. Thus,

$$\models_{\overline{\kappa}} \subseteq \mathfrak{I}(\overline{\kappa}) \times \mathbb{F}_{\Sigma K}$$

That is, in context $\overline{\kappa}$ the interpretation entails a formula in the system Σ^K.

For formulae to be written that have one or more terms referred to in more than one context, term denotations must be rigid. That is, a term's denotation in an interpretation will not vary irrespective of the context in which it is referred to. As such, variable valuations and term denotations are defined relative to the outermost context $\overline{\kappa_\top}$ (similarly to the context-independent declaration of variables in §3.1).

For an interpretation $\nu_{\overline{\kappa}}$, a variable assignment σ for $\Sigma_{\overline{\kappa_\top}}$ is defined as an $\mathcal{S}_{\overline{\kappa_\top}}$-indexed family of functions $\sigma = \{\sigma_S : V_S \to C_S \mid S \in \mathcal{S}_{\overline{\kappa_\top}}\}$. The class of all variable assignments for a context $\overline{\kappa}$ and interpretation $\nu_{\overline{\kappa}}$ is denoted $\mathfrak{V}(\overline{\kappa}, \nu_{\overline{\kappa}})$. The sort of $v \in V$ is unambiguous so $\sigma_S(v)$ is abbreviated $\sigma(v)$.

The denotation of a term t for an interpretation $\nu_{\overline{\kappa}}$ with variable assignment σ is written $[\![t]\!]_\sigma$, defined inductively over the structure of t as :

- $[\![c]\!]_\sigma = c_\nu$ where $c \in C_{\overline{\kappa_\top}}$
- $[\![v]\!]_\sigma = \sigma(v)$ where $v \in V$
- $[\![f(t_1, \ldots, t_n)]\!]_\sigma = f_\nu([\![t_1]\!]_\sigma, \ldots, [\![t_n]\!]_\sigma)$ where $f \in \mathcal{F}_{\overline{\kappa}}$

A well-formed formula that is ill-sorted w.r.t. a context $\overline{\kappa}$ ($\phi \in \mathbb{F}$ but $\phi \notin \mathbb{F}_{\Sigma_{\overline{\kappa}}}$) will be valuated as "meaningless". As such, we interpret such formulae as $\star$ in a three-valued logic ($\{t, f, \star\}$). Ill-sorted terms ($t \notin \mathbb{T}_{\Sigma_{\overline{\kappa}}}(V)$) cause their host predicate to evaluate to $\star$. We adopt Bochvar's system (Bochvar, 1939) which is classical over $\{t, f\}$ for valuating $\neg$ and the propositional connectives but simply propagates $\star$:

p	$\neg p$
t	f
f	t
$\star$	$\star$

$\supset$	t	f	$\star$
t	t	f	$\star$
f	t	t	$\star$
$\star$	$\star$	$\star$	$\star$

Ill-sorted formulae are valuated to $\star$ while well-sorted formulae are interpreted conventionally.

Entailment of a well-formed formula φ by an interpretation $\nu_{\overline{\kappa}}$ (abbreviated here as ν) in context $\overline{\kappa}$ with variable valuation σ, written $\nu \models_{\overline{\kappa}, \sigma} \varphi$ is defined inductively on the structure of φ :

- $\nu \models_{\overline{\kappa}, \sigma} p(t_1, \ldots, t_n)$ iff $\langle [\![t_1]\!]_\sigma, \ldots, [\![t_n]\!]_\sigma \rangle \in p_\nu$
- $\nu \models_{\overline{\kappa}, \sigma} \neg\phi$ iff not $\nu \models_{\overline{\kappa}, \sigma} \phi$
- $\nu \models_{\overline{\kappa}, \sigma} \phi \supset \psi$ iff $\nu \models_{\overline{\kappa}, \sigma} \neg\phi$ or $\nu \models_{\overline{\kappa}, \sigma} \psi$
- $\nu \models_{\overline{\kappa}, \sigma} \mathsf{in}(\kappa_1, \phi)$ iff for each $\nu_i \in \mathfrak{M}(\overline{\kappa} * \kappa_1)$, $\nu_i \models_{\overline{\kappa} * \kappa_1, \sigma} \phi$
- $\nu \models_{\overline{\kappa}, \sigma} \forall v : S \cdot \phi$ iff for each $o \in C_S$, $\nu \models_{\overline{\kappa}, \sigma \oplus [o/v]} \phi$ where $\sigma \oplus [o/v]$ represents the variable assignment mapping σ substituting o for v
- $\nu \models_{\overline{\kappa_2}, \sigma} \phi$ if $\nu \models_{\overline{\kappa_1}, \sigma} \phi$ and $\overline{\kappa_1} \ll^\kappa \overline{\kappa_2}$

$$
\begin{array}{ll}
[\mathsf{P}] & \vdash_{\overline{\kappa}} \phi \qquad \text{where } \phi \text{ is a propositional tautology} \\
[\mathsf{UI}] & \vdash_{\overline{\kappa}} \forall v : S \cdot \phi(v) \supset \phi(t) \qquad \text{where } v \in C_S \text{ and } t \text{ is of a compatible sort} \\
[\mathsf{t}{=}] & \vdash_{\overline{\kappa}} t \doteq t \\
[\mathsf{p}{=}] & \vdash_{\overline{\kappa}} (t_i \doteq t_i') \supset (p(t_1, ..., t_{i-1}, t_i, t_{i+1}, ..., t_n) \supset p(t_1, ..., t_{i-1}, t_i', t_{i+1}, ..., t_n)) \\
[\mathsf{f}{=}] & \vdash_{\overline{\kappa}} (t_i \doteq t_i') \supset (f(t_1, ..., t_{i-1}, t_i, t_{i+1}, ..., t_n) \doteq f(t_1, ..., t_{i-1}, t_i', t_{i+1}, ..., t_n)) \\
[\mathsf{K}] & \vdash_{\overline{\kappa}} \mathsf{in}(\kappa_1, \phi \supset \psi) \supset (\mathsf{in}(\kappa_1, \phi) \supset \mathsf{in}(\kappa_1, \psi)) \\
[\mathsf{EM}] & \vdash_{\overline{\kappa}} \mathsf{in}(\kappa_1, \mathsf{in}(\kappa_2, \phi)) \vee \mathsf{in}(\kappa_1, \neg \mathsf{in}(\kappa_2, \phi)) \\
\\
[\mathsf{BF}] & \vdash_{\overline{\kappa}} \forall v : S \cdot \mathsf{in}(\kappa, \phi) \supset \mathsf{in}(\kappa, \forall v : S \cdot \phi) \\
[\mathsf{RD}] & \vdash_{\overline{\kappa}} t_1 = t_2 \equiv \mathsf{in}(\kappa, t_1 = t_2)
\end{array}
$$

$$
\frac{\vdash_{\overline{\kappa}} \phi \quad \vdash_{\overline{\kappa}} \phi \supset \psi}{\vdash_{\overline{\kappa}} \psi}\ [\mathsf{MP}] \qquad \frac{\vdash_{\overline{\kappa}} \phi \supset \psi(v)}{\vdash_{\overline{\kappa}} \phi \supset (\forall v' : S) \cdot \psi(v')}\ [\mathsf{UG}] \text{ where } v \text{ is not free in } \phi
$$

$$
\frac{\vdash_{\overline{\kappa}} \mathsf{in}(\kappa_1, \phi)}{\vdash_{\overline{\kappa} * \kappa_1} \phi}\ [\mathsf{Enter}] \qquad \frac{\vdash_{\overline{\kappa} * \kappa_1} \phi}{\vdash_{\overline{\kappa}} \mathsf{in}(\kappa_1, \phi)}\ [\mathsf{Exit}] \qquad \frac{\vdash_{\overline{\kappa_1}} \phi \quad \overline{\kappa_1} \ll^{\kappa} \overline{\kappa_2}}{\vdash_{\overline{\kappa_2}} \phi}\ [\mathsf{Sub}\overline{\kappa}]
$$

Figure 3: Inference rules and axioms for reasoning in the system

An interpretation $\nu_{\overline{\kappa}}$ *satisfies* a formula ϕ and is thus a *model* for ϕ iff for all $\sigma \in \mathfrak{V}(\overline{\kappa}, \nu_{\overline{\kappa}})$, $\nu_{\overline{\kappa}} \models_{\overline{\kappa}, \sigma} \phi$. The class of all models for ϕ in $\overline{\kappa}$ is denoted $\mathfrak{M}(\overline{\kappa}, \phi) = \{\nu \mid \nu \models_{\overline{\kappa}} \phi\}$. The $\models$ relation can now be extended to pairs of formulae, that is

$$\phi \models_{\overline{\kappa}} \psi \text{ iff } \mathfrak{M}(\overline{\kappa}, \phi) \subseteq \mathfrak{M}(\overline{\kappa}, \psi)$$

A formula ϕ is *entailed* in context $\overline{\kappa}$ iff ϕ is consistent with all models of context $\overline{\kappa}$ and permissible variable assignments, that is

$$\models_{\overline{\kappa}} \phi \text{ iff } (\forall \nu_{\overline{\kappa}} \in \mathfrak{M}(\overline{\kappa}))(\forall \sigma \in \mathfrak{V}(\overline{\kappa}, \nu_{\overline{\kappa}})) \cdot \nu_{\overline{\kappa}} \models_{\overline{\kappa}, \sigma} \phi$$

The unadorned form of the entailment relation $\models$ is thus defined as the family of $\models_{\overline{\kappa}}$ relations defined over the set of context identifiers K (of the family of signatures Σ^K).

A number of points deserve mention. Within a context, the semantics are completely classical where the formulae of that context do not contain any embedded in subformulae. Where formulae are participants in the embedding and subsequent order-sorting of contexts, a structuring of the space of models is facilitated. Irrespective, the interpretations defined honour the sortal structures of a context scoping as augmented by participation in $\ll^{\kappa}$.

3.3 Reasoning in the System

A precondition for $\vdash_{\overline{\kappa}} \phi$ is that $\phi \in \mathbb{F}_{\Sigma_{\overline{\kappa}}}$, that is ϕ is a well-sorted formula that satisfies the vocabulary and sort specifications of the signature $\Sigma_{\overline{\kappa}}$. Figure 3 presents the inference rules and axioms of our system.

A *proof* of formula ϕ in $\overline{\kappa}$ is a finite sequence of provability relations $(\vdash_{\overline{\kappa_i}} \phi_i)$ terminated by $\vdash_{\overline{\kappa}} \phi$ where every element of the sequence is either an axiom schema instantiation or an immediate consequence of an inference rule application to a previous sequence element. $\vdash_{\overline{\kappa}} \phi$ iff a proof exists.

Due to space restrictions, soundness and completeness results are not provided but are available in the full version of this paper. While a Hilbert-style system has been defined for consistency with the semantic definitions, a natural deduction-style system would be more appropriate for naturally capturing reasoning into and out of embedded contexts, as noted in (McCarthy, 1993; Sharma, 1995).

The axioms [BF] and [RD] state, respectively, that the domains of interpretation for all contexts sequences are equal and that all terms are rigid designators, a term denoting the same object irrespective of context. These assumptions are common to quantified modal logics, of which this system is a variant, but will require addressing (§3.4).

The inclusion of the [Enter] and [Exit] proof rules highlights the reasoning steps of entering and leaving embedded contexts. Application of [Enter] has the effect of "assuming the assumptions embodied in context κ_1" and then reasoning with the unadorned *wff* ϕ. While this might seem to correspond directly to Gentzen-style assumption of κ, such a treatment presumes application of material implication, suggesting a treatment $\mathsf{in}(\kappa, \phi) \triangleq \kappa \supset \phi$. One of the key problems (Sharma, 1995) with this presumption is that for any κ_i and κ_j, $\mathsf{in}([\kappa_i, \kappa_j], \phi) \equiv \mathsf{in}([\kappa_j, \kappa_i], \phi)$ due to the following equivalence :

$$\left.\begin{array}{l} \mathsf{in}([\kappa_i, \kappa_j], \phi) \triangleq (\kappa_i \supset (\kappa_j \supset \phi)) \equiv \neg\kappa_i \vee \neg\kappa_j \vee \phi \\ \mathsf{in}([\kappa_j, \kappa_i], \phi) \triangleq (\kappa_j \supset (\kappa_i \supset \phi)) \equiv \neg\kappa_j \vee \neg\kappa_i \vee \phi \end{array}\right\} \equiv$$

However, we wish distinct context sequences to identify different contexts, irrespective of whether they

contain the same context symbols as the path by which a context is reached should be considered significant (§2). Our treatment of this distinction is consistent with those of (Buvač et al., 1995; Giunchiglia and Traverso, 1995; Nayak, 1994) but not (Attardi and Simi, 1995) where embedded context symbols are treated set-theoretically rather than as an iterated modality.

3.4 Remarks

We only briefly describe the obvious correspondence of our logic to a quantified modal logic. The multi-modal analogues of the axioms and inference rules of Figure 3 would be present. A bijective translation can be defined over all $\phi \in \mathbb{F}_{\Sigma K}$ s.t. $\mathsf{in}(\kappa_1, \phi)$ is translated to $\Box_{\kappa_1}\phi$. $[\kappa_0^1, ..., \kappa_m^1] \ll^\kappa [\kappa_0^2, ..., \kappa_n^2]$ can be translated to the axiom schema $\Box_{\kappa_0^1}...\Box_{\kappa_m^1}\phi \supset \Box_{\kappa_0^2}...\Box_{\kappa_n^2}\phi$. Context-specific signatures can be treated with conventional translations to an unsorted language. Clearly, the system is closed under entailment and so does not improve reasoning efficiency - representational preservation is the purpose of the formalism.

A number of issues will be addressed in future versions of the logic. The assumption of constant domains formalised by [BF] in Figure 3 simplifies the relating of models in different contexts. It is obvious that rather than being interpreted over a constant universe of objects $\mathcal{D}$, different contexts may have non-equal domains of interpretation. One could imagine, for example, there would exist objects meaningful in only one of the contexts $[\kappa_{1996}, \kappa_{netw}]$ and $[\kappa_{1996}, \kappa_{politics}]$.

Such a non-symmetric treatment of contexts' domains requires more careful consideration of the choice of three-valued valuation, however. The bijective correspondence mapping between in and $\Box$ described earlier would have to be augmented with appropriate domain restrictions for variables within their quantifying scopes. Further, relationships between contexts such as captured by $\ll^\kappa$ will place constraints over the domains of the respective contexts. §4 illustrates such an example where it is clear the contexts' domains are strongly related.

Terms as rigid designators preclude a convenient treatment of anaphoric references such as the representation of pronouns in natural language where denotations can vary according to context. Our principal concern in this work, however, is the flexible representation of ontologies and, as such, see no immediate need to address this limitation.

Sorts should be mirrored by corresponding unary predicates so that sorts may be intensionally defined, similarly to *defined concepts* in KL-ONE style description logics (Woods and Schmolze, 1992). In conventional quantified order-sorted systems such as §3, the sortal structure is used to enforce well-sortedness of terms. That is, a sort is not a *wff* and so cannot have definitional constraints stated for it. The approach taken by (Beierle et al., 1992) which we plan to adopt involves reflecting the set of sorts in the set of unary predicate symbols of sort $\top$.

For example, in our system it is not possible to state that a sort *Aus_Repeater* specialises *Repeater* s.t. the equality $Manufactured\_in(x) = \text{`}Australia\text{'}$ is satisfied. If, however, sorts were reflected in the unary predicates, such a requirement could be formalised as

$$\forall x : Repeater \cdot Aus\_Repeater(x) \equiv Manufactured\_in(x) = \text{`}Australia\text{'}$$

The modification to our system would correspond to stating $\mathcal{P}^{\top}_{\overline{\kappa}} \supseteq \mathcal{S}_{\overline{\kappa}}$ and relaxing the requirement that $\mathcal{P}_{\overline{\kappa}}$ and $\mathcal{S}_{\overline{\kappa}}$ are disjoint for all context sequences $\overline{\kappa}$. Corresponding extensions in the semantics and inference system would be made (extending unification / resolution in (Beierle et al., 1992)), facilitating "closely coupled taxonomic reasoning". Such an approach would be particularly appropriate with our context mechanism for situations where a subsort is intensionally defined w.r.t. a parent sort in terms of context-specific properties. We believe these issues will have ramifications for integrating contexts and description logic systems.

It would be useful to relax the well-sortedness criteria to allow lazy or "soft" sort evaluation so that lifting rules can be conveniently written in terms of the most general sort $\top$.

4 Example revisited

The example of §1.1 could be modelled by a simple context configuration as illustrated in Figure 4. Assuming the contexts are situated in a context $[\kappa_{lan}]$, we can write $\{([\kappa_{lan}, \kappa_{netw}], [\kappa_{lan}, \kappa_{filt}]), ([\kappa_{lan}, \kappa_{netw}], [\kappa_{lan}, \kappa_{resp}])\} \subseteq \ll^\kappa$, indicating that in the context of LANs (κ_{lan}), κ_{filt} and κ_{resp} are specialisations of a more general networking context κ_{netw}. (We shall drop the $[\kappa_{lan}, ..]$ for the rest of this section for brevity.)

The principal sorts would be declared by $\{Bridge, Repeater\} \subseteq \mathcal{S}_{\kappa_{netw}}$ although neither would participate in $\ll_{\kappa_{netw}}$. $\Sigma_{\kappa_{filt}}$ and $\Sigma_{\kappa_{resp}}$ could introduce context-specific properties and sort orderings as follows :

$$\mathcal{S}_{\Delta_{filt}} = \{DevX\}$$
$$\ll_{\Delta_{filt}} = \{(Repeater, Bridge), (Repeater, DevX)\}$$
$$\mathcal{S}_{\Delta_{resp}} = \{DevY\}$$
$$\ll_{\Delta_{filt}} = \{(Bridge, Repeater), (Bridge, DevY)\}$$

The properties in Figure 1 could be modelled as unary predicates with sorted terms, exploiting the respective sortal structures established in $\Sigma_{\kappa_{netw}}$, $\Sigma_{\kappa_{filt}}$ and

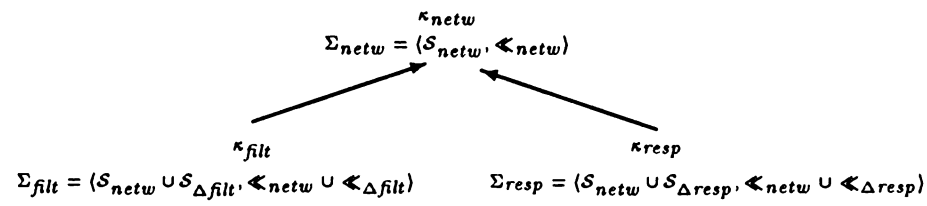


Figure 4: Theory structure showing sorts and orders in inheriting contexts

$\Sigma_{\kappa resp}$:

$$\{A, B\} \subseteq \mathcal{P}^{\top}_{\kappa_{netw}}$$
$$\{Filtering\text{-}property\} \subseteq \mathcal{P}^{Repeater}_{\kappa_{filt}}$$
$$\{Response\text{-}property\} \subseteq \mathcal{P}^{Bridge}_{\kappa_{resp}}$$

These structures clearly preserve the original subsumption orderings and enables sorts (*DevX*,*DevY*) and properties (*Filtering-property*,*Response-property*) to be restricted to the signatures of the appropriate contexts. In the context κ_{netw}, the sorts *Bridge* and *Repeater* are "free" w.r.t. subsumption commitments while reference to specialising contexts κ_{filt} and κ_{resp} formalises the ontology's subsumption relationships.

Figure 5 illustrates a slightly more complex variant on §1.1. A distinction is introduced between the signatures and formulae of the context κ_{netw} from the perspectives of κ_{1986} and κ_{1996}. The distinctions of the filtering and response-time contexts κ_{filt} and κ_{resp} are deemed relevant in κ_{1996} but not for κ_{1986}. The context sequence $[\kappa_{lan}, \kappa_{netw}]$ can be seen intuitively as capturing the invariant (relative to $[\kappa_{lan}]$) properties of more deeply embedded κ_{netw}s. This effectively provides a prototype signature and set of formulae to $[\kappa_{lan}, \kappa_{1986}, \kappa_{netw}]$ and $[\kappa_{lan}, \kappa_{1996}, \kappa_{netw}]$, written

$$\{([\kappa_{lan}, \kappa_{netw}], [\kappa_{lan}, \kappa_{1986}, \kappa_{netw}]),$$
$$([\kappa_{lan}, \kappa_{netw}], [\kappa_{lan}, \kappa_{1996}, \kappa_{netw}])\} \subseteq \ll^{\kappa}$$

A more general schema-based approach will allow the populating of $\ll^{\kappa}$ for prototypical context symbols (e.g. κ_{netw}) within a context scope (e.g. $[\kappa_{lan}]$).

Consider that $[\kappa_{lan}, \kappa_{1996}, \kappa_{interconn}]$ identifies a context defining a simple ontology for internetworking, representing notions from κ_{netw} such as **Bridge** and **Repeater** more generally as **Connective**. The *less_abstract_than* relationship in Figure 5 between the context sequences $[\kappa_{lan}, \kappa_{1996}, \kappa_{netw}]$ and $[\kappa_{lan}, \kappa_{1996}, \kappa_{interconn}]$ captures an abstraction relationship where the latter contains a more abstract representation of the formulae of the former with lifting rules relating the formulae of the respective contexts. This context relationship corresponds to the idea of a "theorem-increasing" abstraction (TI-abstraction) in the classification scheme presented in (Giunchiglia and Walsh, 1992). Such relationships are accommodated in our framework as higher-level relationships over context sequences (extrinsic to and layered above the semantics of §3) that may be exploited by a reasoner during problem-solving.

These ideas form the basis for context spaces and are similar to the approach of reasoning with multiple models of a system in the qualitative reasoning community (Addanki et al., 1991). More extensive application of embedded contexts and signature prototyping are available in (Sharma, 1996).

5 Related Work

McCarthy first proposed the importance of formalised contexts for artificial intelligence in (McCarthy, 1990) and subsequently in (McCarthy, 1993) as a possible solution to the problems of generality. Buvač (Buvač et al., 1995) defines a propositional logic of contexts with a modal semantics, introducing the **ist** modality, context sequences for context identification and context-specific propositional symbols. Nayak restricts his propositional modal system (Nayak, 1994) to just a flat space of contexts and models context inheritance through axiom schema (c.f. §3.4).

In (Buvač, 1996), Buvač sketches a first order system that allows context variables to be quantified over $(\forall \kappa \cdot \phi)$. Context-specific vocabularies are disallowed, however, as it is not possible to do a static analysis of the vocabulary of a *wff* as was possible in §3.1 and (Buvač et al., 1995) as the context variable needs to be bound to a context symbol before a vocabulary analysis can be done on the *wff*. For the purposes of flexible ontology specification, it does not seem that the complications of quantified contents are justified.

Other notable approaches for reasoning with contexts include the syntactic approaches of Attardi (Attardi and Simi, 1995) and Giunchiglia (Giunchiglia and Traverso, 1995), although the former expends considerable effort avoiding paradoxes due to Montague's diagonalisation theorem. Prominent in the linguistic community is Barwise's work on situation semantics (Seligman and Moss, 1996) which, while recognising specialising situation objects similarly to §2, attempts to deal with situations as first-class and self-referential objects. An extensive survey on formalising contexts can be found in (Sharma, 1995).

While we have stated that our essentially modal ap-

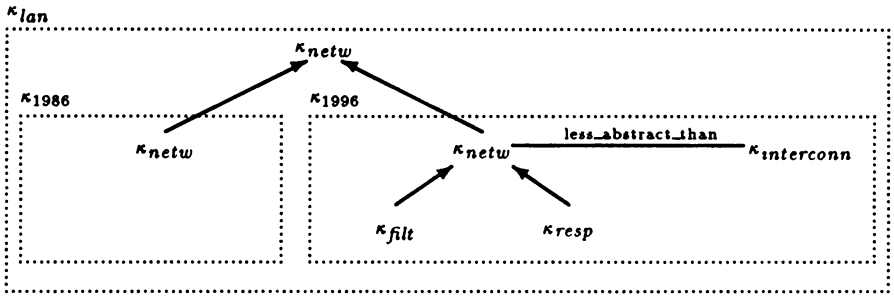


Figure 5: Example of order-sorting across embedded contexts

proach is more appropriate to our goals, the work should not be confused with simply adding intensional modalities such as belief. Our axiomatisation does not include any generalised variant on the axioms common to logics of belief of positive and negative introspection, respectively :

[4] $\Box\phi \supset \Box\Box\phi$
[5] $\Diamond\phi \supset \Box\Diamond\phi$

Our system enforces no such reflection of the structures of context sequences in the relationship between the signatures or models of the contexts, other than via $\ll^{\kappa}$. Rather, we are interested in generalising the sortal structure and its subsequent interpretation.

Wille's work on triadic concept analysis (Wille, 1992) is worthy of note as it introduces a third dimension to sort lattices to form a trilattice with contexts. However, the system is propositional with no deduction component provided other than reasoning with the various subsumption relationships in the trilattice.

Dimensionality of inheritance as discussed often mistakenly motivates non-monotonic inheritance formalisms. The requirement to reduce multiple dimensions of inheritance to a single concept space is often encoded into policies where a certain property may hold for a concept but the alternate perspective or situation is denoted by a specialising concept for which the corresponding property is treated as an override. Our experience suggests that the application of formalised contexts for clarifying inheritance and subsumption in situations conventionally understood as necessitating non-monotonic default reasoning is worthy of further investigation. Some initial work to this end has been carried out in (Giunchiglia and Weyhrauch, 1988).

6 Conclusions

Conventional treatments of concept taxonomies represent a single subsumption partial order over a set of concepts. A chosen partial order w.r.t. certain properties may not be motivated by the desired ordering for the inheritance of other properties. Given that perspectives or, more generally, contexts can decide properties modelled, order-sorting is clearly context-sensitive. A modeller often wishes these subsumption relationships to be preserved as such information lost when describing an ontology could be significant, particularly if the ontology is expected to have application beyond one task.

The formal system presented in this paper enables originating contexts of sorts to be preserved. Contexts themselves may be stated relative to other contexts, providing the basis for context spaces upon which large, multi-function ontologies can be constructed whose formulae may have been acquired from different perspectives or contexts. Contexts provide a more flexible mechanism for structuring theories than do modularity notions as relationships can be stated between formulae in contexts.

Acknowledgements

The author would like to thank Robert Colomb and Peter Robinson as well as the anonymous reviewers for their comments and suggestions. The work reported in this paper has been funded in part by the Cooperative Research Centres Programme through the Department of the Prime Minister and Cabinet of the Commonwealth Government of Australia.

References

Addanki, S., Cremonini, R., and Penberthy, J. S. (1991). Graph of models. *Artificial Intelligence*, 51:145–177.

Attardi, G. and Simi, M. (1995). A formalization of viewpoints. *Fundamenta Informaticæ*, 23(3):149–173.

Beierle, C., Hedtstück, U., Pletat, U., Schmitt, P. H., and Siekmann, J. (1992). An order-sorted logic for knowledge representation systems. *Artificial Intelligence*, 55:149–191.

Bochvar, D. A. (1939). Ob odnom tréhznaćnom isčislénii i égo priménénii k analizu paradoksov klassičéskogo rasširénnogo funkcional'nogo isčisléniá. *Matématičeskij Sbornik*, 4:287–308. (On a 3-valued

logical calculus and its application to the analysis of contradictions).

Buvač, S. (1996). Quantification logic of context. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96), Portland*, pages 600–606.

Buvač, S., Buvač, V., and Mason, I. A. (1995). Metamathematics of contexts. *Fundamenta Informaticæ*, 23(3):263–301.

Giunchiglia, E. and Traverso, P. (1995). A multi-context architecture for formalizing complex reasoning. *International Journal of Intelligent Systems*, 10:501–539.

Giunchiglia, F. and Walsh, T. (1992). A theory of abstraction. *Artificial Intelligence*, 56(2-3):323–390.

Giunchiglia, F. and Weyhrauch, R. W. (1988). A multi-context monotonic axiomatization of inessential non-monotonicity. In Nardi, D. and Maes, P., editors, *Meta-Level Architectures and Reflection*, pages 271–285. North Holland. also available as IRST Technical Report #91-0004.

McCarthy, J. (1990). Generality in artificial intelligence. In Lifschitz, V., editor, *Formalizing common sense: papers by John McCarthy*, pages 226–236. Ablex Press, Norwood, New Jersey. Turing Award Lecture (1971), reprinted in CACM 30(12):1030–1035, December 1987.

McCarthy, J. (1993). Notes on formalizing context. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93), Chambery*, pages 555–560.

Mittal, S. and Frayman, F. (1987). Making partial choices in constraint reasoning problems. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87), Seattle*, pages 631–636.

Nayak, P. P. (1994). Representing multiple theories. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94), Seattle*, pages 1154–1160.

Quine, W. V. (1969). Ontological relativity. In *Ontological Relativity and other essays*, pages 26–68. Columbia University Press, New York.

Seligman, J. and Moss, L. (1996). Situation theory. In van Benthem, J. and ter Meulen, A., editors, *Handbook of Logic and Linguistics*. Elsevier. *to appear*.

Sharma, N. (1995). On formalising and reasoning with contexts. Technical Report 352, Department of Computer Science, The University of Queensland. *submitted*.

Sharma, N. (1996). Representing context and reasoning with multiple theories - a network diagnosis / configuration scenario. Technical report, Department of Computer Science, The University of Queensland.

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

Woods, W. A. and Schmolze, J. G. (1992). The KL-ONE family. *Computers and Mathematics with Applications*, 23(2–5):133–177.

Description Logics: Reasoning Techniques

Finite Model Reasoning in Description Logics

Diego Calvanese
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, I-00198 Roma, Italy
calvanese@dis.uniroma1.it

Abstract

For the basic Description Logics reasoning with respect to finite models amounts to reasoning with respect to arbitrary ones, but finiteness of the domain needs to be considered if expressivity is increased and the finite model property fails. Procedures for reasoning with respect to arbitrary models in very expressive Description Logics have been developed, but these are not directly applicable in the finite case. We first show that we can nevertheless capture a restricted form of finiteness and represent finite modeling structures such as lists and trees, while still reasoning with respect to arbitrary models. The main result of this paper is a procedure to reason with respect to finite models in an expressive Description Logic equipped with inverse roles, cardinality constraints, and in which arbitrary inclusions between concepts can be specified without any restriction. This provides the necessary expressivity to go beyond most semantic and object-oriented Database models, and capture several useful extensions.

1 INTRODUCTION

From the start of the discipline, researchers working in Knowledge Representation (KR), have aimed at augmenting the expressivity of the formalisms used for representing structured knowledge, without compromising the computational properties of the procedures adopted for reasoning about it. This research has paralleled the one in Databases (DBs), where expressive data models have been developed, aimed at representing the data manipulated by commercial applications. There is a strong similarity between all these approaches, in which the domain knowledge is represented in form of a schema by defining classes, which denote subsets of the domain, and by specifying various types of relations between classes, which establish their structural properties (Calvanese, Lenzerini, & Nardi, 1994; Borgida, 1995). In this context it is fundamental to provide methods for reasoning about such specifications in order to detect inconsistencies and hidden dependencies which arise from the specified constraints. This is precisely the objective of all KR systems in the style of KL-ONE, and of Description Logics (DL) introduced for their formalization. It is gaining an increased importance also in DBs, both to support the phase of analysis and design of DB systems, and in the process of query answering to perform for example semantic query optimization.

However, while in DBs it is usually assumed that the underlying domain is finite, reasoning in DLs is performed without making such hypothesis. This seems to be in contrast with the purpose of a KR system to represent real world structures, which are inherently finite. It can however be justified by observing that most DLs studied so far have the finite model property, stating that a consistent schema (or KB) always admits a model with finite domain. This implies that one does not need to take special care about finiteness of the domain, since it can always be assumed. Unfortunately, the finite model property does not hold for more expressive logics, as those studied for example in (De Giacomo & Lenzerini, 1994a; Buongarzoni, Menghini, Salis, Sebastiani, & Straccia, 1995). In particular, this happens also for the logics which include inverse roles and functionality of roles, and which therefore have sufficient expressivity to represent semantic and object-oriented DB models (Calvanese et al., 1994). For those logics it becomes important to provide specific reasoning methods for both the finite and the unrestricted case.

While for unrestricted reasoning, techniques have been developed to deal with very expressive DLs (De Giacomo & Lenzerini, 1995), decidability of reasoning with respect to finite models is still open for important cases. In particular, there is no known method capable of handling schemata in which classes can be defined by necessary and sufficient conditions, in a logic featuring the constructs typical of DB models. Such methods are however necessary if one wants to reason for example on views in object-oriented models. Moreover, excessively limiting expressivity goes against the spirit of capturing in the schema as much as possible of the semantics of the modeled reality. Only an expressive formalism equipped with sound and complete reasoning procedures provides the user with the necessary tools to reason about relevant aspects of the domain he needs to represent. In this paper we present such formalisms in which different forms of finiteness can be captured.

On one hand, we show that while keeping the maximum expressivity, one can add to the language a *well-founded* constructor which allows one to impose well-foundedness and therefore finiteness of certain structures, without necessarily requiring the whole domain to be finite. By means of the well-founded constructor it becomes possible to define classes that represent modeling structures such as lists and trees, and reason about them like about any other class in the schema. This represents a significant improvement with respect to traditional data models, where such modeling structures, if present at all, are ad hoc additions that require a special treatment by the reasoning procedures (Cattell, 1994). The known methods for reasoning with respect to arbitrary models in expressive DLs are based on a correspondence between DLs and Propositional Dynamic Logics (PDLs) and exploit the powerful reasoning methods developed for PDLs (Schild, 1991; De Giacomo & Lenzerini, 1994a). We show that the correspondence can in fact be extended to handle also well-foundedness, and present a reasoning technique for the resulting PDL.

On the other hand, we develop a method to reason with respect to finite models on schemata built using a very expressive DL, and show its decidability in deterministic double exponential time. The DL we consider includes besides number restrictions and inverse roles also general negation of concept expressions. The schemata are of the most general form, in which one can express arbitrary inclusions between complex expressions without any restriction at all. This makes our language not only capable of capturing the structural properties of class-based representation formalisms used in DBs, but provides also the necessary expressivity to subsume the terminological component of most existing concept based systems.

The rest of the paper is organized as follows: In Section 2 we present the representation formalism we adopt, in Section 3 we discuss the issues related to reasoning about finite modeling structures and in finite models and in Sections 4 and 5 we analyze the reasoning methods used in both contexts. In Section 6 we outline possible directions for future work.

2 THE REPRESENTATION FORMALISM

In this section, we present the family of class-based formalisms we deal with. These formalisms exploit Description Logics for the definition of schemata, and the logics we define extend the well known logics of the $\mathcal{AL}$-family (Donini, Lenzerini, Nardi, & Nutt, 1991a) by several expressive constructs.

2.1 SYNTAX AND SEMANTICS OF DESCRIPTION LOGICS

The basic elements of *Description Logics* (*DLs*) are *concepts* and *roles*, which denote classes and binary relations, respectively. Complex concept and role expressions are built, starting from a set of *concept* and *role names*, by applying certain constructors. Each DL is characterized by the set of allowed constructors. Table 1 shows the concept and role forming constructors we consider in this paper. The first column specifies the name of each constructor and the second column specifies a letter that is used to denote each constructor when naming a logic. The basic logic we consider is $\mathcal{AL}$, which contains only the constructors to which no letter is associated. All other logics we deal with include the constructors of $\mathcal{AL}$. Each logic is named with a string that starts with $\mathcal{AL}$ and includes the letters that are associated to the additional constructors in the logic.

The syntax of each constructor in concept and role expressions is shown in the third column of the table. Concept names are denoted by C and complex concept expressions by E. Role names and complex role expressions are denoted by P and R, respectively. m denotes a positive and n a nonnegative integer.

Concepts are interpreted as subsets of a domain and roles as binary relations over that domain. More precisely, an *interpretation* $\mathcal{I} := (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a set $\Delta^{\mathcal{I}}$ (the *domain* of $\mathcal{I}$) and a function $\cdot^{\mathcal{I}}$ (the *interpretation function* of $\mathcal{I}$) that maps every concept name C to a subset $C^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, every role name P to a sub-

Constructor Name		Syntax	Semantics
concept name		C	$C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
top		$\top$	$\Delta^{\mathcal{I}}$
atomic negation		$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
conjunction		$E_1 \sqcap E_2$	$E_1^{\mathcal{I}} \cap E_2^{\mathcal{I}}$
universal quantification		$\forall R.E$	$\{o \mid \forall o' : (o,o') \in R^{\mathcal{I}} \rightarrow o' \in E^{\mathcal{I}}\}$
unqualified existential quantification		$\exists R$	$\{o \mid \exists o' : (o,o') \in R^{\mathcal{I}}\}$
existential quantification	$\mathcal{E}$	$\exists R.E$	$\{o \mid \exists o' : (o,o') \in R^{\mathcal{I}} \wedge o' \in E^{\mathcal{I}}\}$
disjunction	$\mathcal{U}$	$E_1 \sqcup E_2$	$E_1^{\mathcal{I}} \cup E_2^{\mathcal{I}}$
general negation	$\mathcal{C}$	$\neg E$	$\Delta^{\mathcal{I}} \setminus E^{\mathcal{I}}$
number restrictions	$\mathcal{N}$	$\exists^{\geq m} R$	$\{o \mid \sharp\{o' \mid (o,o') \in R^{\mathcal{I}}\} \geq m\}$
		$\exists^{\leq n} R$	$\{o \mid \sharp\{o' \mid (o,o') \in R^{\mathcal{I}}\} \leq n\}$
qualified number restrictions	$\mathcal{Q}$	$\exists^{\geq m} R.E$	$\{o \mid \sharp\{o' \mid (o,o') \in R^{\mathcal{I}} \wedge o' \in E^{\mathcal{I}}\} \geq m\}$
		$\exists^{\leq n} R.E$	$\{o \mid \sharp\{o' \mid (o,o') \in R^{\mathcal{I}} \wedge o' \in E^{\mathcal{I}}\} \leq n\}$
well-founded	$\mathcal{W}$	$wf(R)$	$\{o_0 \mid \forall o_1, o_2, \ldots \text{(ad infinitum)}\ \exists i \geq 0 : (o_i, o_{i+1}) \notin R^{\mathcal{I}}\}$
role value map	$\mathcal{V}$	$(R_1 \subseteq R_2)$	$\{o \mid \{o' \mid (o,o') \in R_1^{\mathcal{I}}\} \subseteq \{o' \mid (o,o') \in R_2^{\mathcal{I}}\}\}$
role name		P	$P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
inverse	$\mathcal{I}$	R^-	$\{(o,o') \mid (o',o) \in R^{\mathcal{I}}\}$
union	$\mathcal{R}$	$R_1 \cup R_2$	$R_1^{\mathcal{I}} \cup R_2^{\mathcal{I}}$
concatenation	$\mathcal{R}$	$R_1 \circ R_2$	$R_1^{\mathcal{I}} \circ R_2^{\mathcal{I}}$
reflexive transitive closure	$\mathcal{R}$	R^*	$(R^{\mathcal{I}})^*$
identity	$\mathcal{R}$	$id(E)$	$\{(o,o) \mid o \in E^{\mathcal{I}}\}$
difference	$\mathcal{D}$	$R_1 \setminus R_2$	$R_1^{\mathcal{I}} \setminus R_2^{\mathcal{I}}$

Table 1: Syntax and semantics of the concept and role forming constructors.

set $P^{\mathcal{I}}$ of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and concept and role expressions according to the last column of Table 1[1].

Most of the constructors we have presented are well known in DLs, and have a correspondence in modeling constructs used both in Frame Based Systems and in DB models. This is not the case for the constructor *wf*, called *well-founded*, which has rarely been considered in KR, but which proves particularly useful for expressing modeling structures that are inherently finite.

2.2 $\mathcal{AL}$-SCHEMATA

Using concept expressions of a DL, intensional knowledge can be specified through schemata. Given an $\mathcal{AL}$-DL $\mathcal{L}$, an *$\mathcal{L}$-schema* is a triple $\mathcal{S} := (\mathcal{C}, \mathcal{P}, \mathcal{T})$, where $\mathcal{C}$ is a finite set of concept names, $\mathcal{P}$ is a finite set of role names, and $\mathcal{T}$ is a finite set of *assertions*. Each such assertion has one of the forms

$$C \doteq E \quad \textit{(primitive concept specification)}$$
$$C \doteq E \quad \textit{(concept definition)}$$

where $C \in \mathcal{C}$ and E is a concept expression of $\mathcal{L}$ involving only names of $\mathcal{C} \cup \mathcal{P}$. In the following, when not specified otherwise, we assume that $\mathcal{S} := (\mathcal{C}, \mathcal{P}, \mathcal{T})$, and that $\mathcal{C}$ and $\mathcal{P}$ are the sets of concept and role names that appear in $\mathcal{T}$.

Primitive concept specifications are used to specify necessary conditions for an object to be an instance of a concept, while concept definitions specify both necessary and sufficient conditions. More formally, an interpretation $\mathcal{I}$ *satisfies* an assertion $C \doteq E$ if $C^{\mathcal{I}} \subseteq E^{\mathcal{I}}$, and it satisfies an assertion $C \doteq E$ if $C^{\mathcal{I}} = E^{\mathcal{I}}$. An interpretation that satisfies all assertions in a schema $\mathcal{S}$ is called a *model* of $\mathcal{S}$. A *finite model* is a model with finite domain. A concept expression E is *(finitely) consistent* in $\mathcal{S}$ if $\mathcal{S}$ admits a (finite) model $\mathcal{I}$ such that $E^{\mathcal{I}} \neq \emptyset$, and E_1 is *(finitely) subsumed* by E_2 in $\mathcal{S}$ if $E_1^{\mathcal{I}} \subseteq E_2^{\mathcal{I}}$ for every (finite) model $\mathcal{I}$ of $\mathcal{S}$.

Both in KR and in DB models several different assumptions on the form of a schema are made, either implicitly or explicitly:

1. For each concept name there is at most one assertion in which that name appears in the left-hand side.
2. The schema contains no cycles[2].

[1] $\sharp S$ denotes the cardinality of a set S.

[2] See (Nebel, 1991) for a formal definition of cycle in a schema.

3. The schema contains only primitive concept specifications (*primitive schema*).

Assumption 1 corresponds to the natural requirement that all the information concerning a class is contained in one place, which contributes to a better structuring of the represented knowledge (Nebel, 1990; Baader, 1990; Lecluse & Richard, 1989; Bergamaschi & Sartori, 1992). It may however be too restrictive, especially in those cases where one wants to state additional constraints that are not specific to a certain class.

Assumption 2 of acyclicity (which is meaningful only in the presence of assumption 1) is made in most existing concept-based KR systems, although imposing this condition strongly limits the expressive power of the system (Nebel, 1990). Many real world concepts can in fact be expressed naturally only in a recursive way and thus through cycles. The reason for not admitting cycles is twofold. On one hand cycles greatly increase the computational complexity of reasoning on a schema (Baader, 1990; Calvanese, 1996a), and on the other hand, there is still no agreement in the field on which semantics to adopt in their presence. Besides *descriptive semantics*, which is the semantic specification described above, so called *fixpoint semantics* have been considered (Nebel, 1991). Descriptive semantics, which is the one we use, has been advocated to be the one which gives the most intuitive results (Nebel, 1991; Buchheit, Donini, Nutt, & Schaerf, 1995), and it is also the only one that can be adopted for the most general type of schemata, called *free*, in which none of the three conditions above is required to hold[3]. (Baader, 1990) argues that from a constructive point of view greatest fixpoint semantics should be preferred. Least fixpoint semantics, on the other hand, seems the most appropriate for the definition of inductive structures. We will see in Section 3 that we can obtain a similar result by using the well-founded constructor together with descriptive semantics. We remark that recently there have been also proposals for an integration of the different types of semantics by including fixpoint constructors in the logic (Schild, 1994; De Giacomo & Lenzerini, 1994b) and interpreting cycles with descriptive semantics.

Assumption 3 is sometimes made in concept-based KR systems (Buchheit et al., 1995) and it is usually implicit in DB models, in which the classes are specified only through necessary conditions (Calvanese et al., 1994). Although it strongly limits expressivity, reasoning on primitive schemata is nevertheless intractable even in the simplest setting, where we use only the constructors of $\mathcal{AL}$ (which are common to almost all representation formalisms) and the schema is acyclic (Buchheit et al., 1995). In fact assumption 3 reduces worst case complexity of reasoning on a schema only if relatively weak languages are adopted (Calvanese, 1996a).

[3]The expressivity of free schemata defined in this way is equivalent to the one of schemata constituted by *free inclusion assertions*, which have the form $E_1 \doteq E_2$, where both E_1 and E_2 are arbitrary concept expressions with no restriction at all.

3 FROM FINITE MODELING STRUCTURES TO FINITE MODELS

Both in KR and in DBs representation formalisms have been developed which offer powerful structuring capabilities and procedures to reason about a specification. However, while in DBs the common assumption is that the modeled domain is finite, this has seldom been an issue in KR in general and in DLs in particular, where the developed reasoning methods do not take care of finiteness. For most of the DLs that have been considered till now this does not represent a limitation, neither for reasoning on concept expressions (Donini et al., 1991a; Donini, Lenzerini, Nardi, & Nutt, 1991b) nor on schemata (Nebel, 1991; Buchheit et al., 1995), since for these formalisms the *finite model property* holds. It states that if a schema (or concept expression) admits a model, then it also admits one with a finite domain, and therefore reasoning with respect to unrestricted models amounts to reasoning with respect to finite ones. This does not hold anymore for the more expressive logics studied recently, which include inverse roles and functionality of roles, since these constructors in combination with cycles in the schema cause the finite model property to fail to hold.

Example 1 Consider the following schema:

$$\texttt{Guard} \doteq \exists \texttt{shields}.\texttt{Guard} \sqcap \exists^{\leq 1}\texttt{shields}^{-}$$
$$\texttt{FirstGuard} \doteq \texttt{Guard} \sqcap \exists^{\leq 0}\texttt{shields}^{-}$$

Each **Guard** shields at least some guard, and is shielded by at most one individual. A **FirstGuard** is a guard whom nobody shields. It is easy to see that the existence of a **FirstGuard**forces the existence of an infinite sequence of guards, each one shielding the next. Therefore **FirstGuard** is consistent but not finitely consistent. ∎

For other examples, see the logics in (De Giacomo & Lenzerini, 1994a; Calvanese et al., 1994), or the one in (Buongarzoni et al., 1995), where not cycles but

singleton concepts are used to specify concepts that are consistent only in an infinite domain.

For the most expressive DLs (and in particular for those that lack the finite model property), methods for reasoning with respect to arbitrary models have been developed, which are based on a correspondence between DLs and Propositional Dynamic Logics (PDLs). PDLs are formalisms specifically designed for reasoning about program schemes (Kozen & Tiuryn, 1990), and their correspondence with DLs was described first in (Schild, 1991) and extended to more expressive logics in (De Giacomo & Lenzerini, 1994a). The correspondence allows one to reduce concept consistency to satisfiability of a formula of some PDL, and to use the advanced methods developed for PDLs to solve this latter task. These methods exploit the fundamental *tree model property*, shared by all PDLs, which states that every satisfiable formula admits particular models that have the form of an (infinite) tree of bounded degree. If the domain has to be finite, however, the existence of tree-like models is not guaranteed, and the known reasoning methods are not applicable.

It is nevertheless possible to express restricted forms of finiteness, even in those logics that lack the finite model property, by imposing conditions that force certain structures in the domain to be well-founded, without assuming the whole interpretation domain to be finite. This allows us to deal with finite modeling structures, while exploiting the powerful techniques for reasoning with respect to unrestricted models. Well-foundedness can be imposed by using $wf(R)$, which expresses that there is no infinite sequence of objects, connected one to the next by means of role R. In terms of PDLs, $wf(R)$ corresponds to a negated repeat formula $\neg\Delta(R)$ (Streett, 1982), which is in turn equivalent to the least fixpoint expression[4] $\mu X.\forall R.X$ (Kozen, 1983). Thus we can use such constructor to express finite structures that admit a least-fixpoint definition. We illustrate this on the example of binary trees.

Example 2 We can define a binary tree inductively as the least set *bin-tree* such that:

- a node having no successors is a *bin-tree*,
- a node whose successors are all *bin-trees*, is a *bin-tree*,

and where additionally a *node* is constrained to have at most one predecessor and at most two successors (plus possibly additional properties such as a label).

The inductive part of the definition can be captured by the least fixpoint expression

$$\mu X.((\mathbf{Node} \sqcap \exists^{\leq 0}\mathbf{edge}) \sqcup (\mathbf{Node} \sqcap \forall \mathbf{edge}.X))$$

which is equivalent to

$$\mu X.(\mathbf{Node} \sqcap \forall \mathbf{edge}.X).$$

Thus, the following (non-recursive) schema reflects the full definition of binary tree, where the first assertion in the schema captures the local properties of nodes:

$$\begin{aligned} \mathbf{Node} &\doteq \exists^{\leq 2}\mathbf{edge} \sqcap \exists^{\leq 1}\mathbf{edge}^{-} \sqcap \exists \mathbf{label}.\ldots \\ \mathbf{BinTree} &\doteq \mu X.(\mathbf{Node} \sqcap \forall \mathbf{edge}.X) \end{aligned} \quad (1)$$

It is possible to show that, for every model $\mathcal{I}$ of a schema $\mathcal{S}$ containing the assertions above, the set of objects that are instances of **BinTree** remains the same if we replace assertion 1 by the following:

$$\mathbf{BinTree} \doteq \mathbf{Node} \sqcap \forall \mathbf{edge}.\mathbf{BinTree} \sqcap wf(\mathbf{edge}) \quad (2)$$

Notice that this assertion is recursive and makes use of the well-founded constructor to achieve the same effect as the least-fixpoint expression. In particular, given an interpretation $\mathcal{I}$ of $\mathcal{S}$ that interprets all concept and role names in $\mathcal{S}$ except **BinTree**, there is a unique way to extend $\mathcal{I}$ to **BinTree** such that it satisfies assertion 2. The extension of **BinTree** is exactly the set of objects that are instances of the least-fixpoint expression on the right hand side of assertion 1. Therefore assertion 2, despite being recursive, represents a proper definition. ∎

By proceeding in a similar way, most data structures encountered in Computer Science which admit an inductive definition, such as lists, trees, and directed acyclic graphs, can be modeled in a DL including the well-founded constructor. Notice that in general to this end both inverse roles and number restrictions are indispensable. For this reason we can not directly make use of fixpoint constructors as in the DLs presented in (Schild, 1994; De Giacomo & Lenzerini, 1994b), since decidability is still open for those logics including fixpoints in combination with inverse roles.

Another important use of the well-founded constructor is for the definition of well-founded relations, of which the *part-of* relation is a notable example with great practical relevance (Artale, Franconi, Guarino, & Pazzi, 1996; Sattler, 1995). Transitivity of the part-of

[4] A least fixpoint expression $\mu X.f(X)$, where X is a concept variable and f is a monotone operator, is interpreted as the least set $X^{\mathcal{I}}$ of objects in $\Delta^{\mathcal{I}}$ satisfying $X^{\mathcal{I}} = (f(X))^{\mathcal{I}}$. See (Schild, 1994; De Giacomo & Lenzerini, 1994b) for more details.

relation can be captured by taking the transitive closure of a `basic_part_of` role, while asymmetricity and finiteness (i.e. well-foundedness) are imposed by an assertion of the form $\top \doteq wf(\texttt{basic\_part\_of})$. Notice also that by means of role value maps on role names we can capture different specializations of the part-of relation.

The constructors that cause the loss of the finite model property are all necessary in order to capture the characteristics of the most important representation formalisms used both in KR and in DBs (Calvanese et al., 1994). In fact, neither the Entity-Relationship model nor expressive object-oriented models have the finite model property, and since the assumption of a finite domain is essential in this context, specialized methods capable of reasoning in these formalisms with respect to finite models are needed. This seems to be more difficult than reasoning with respect to arbitrary ones. As already noticed, if the domain has to be finite the tree model property does not hold, and the correspondence with PDLs cannot be exploited. In fact, decidability of finite concept consistency and finite subsumption for more expressive formalisms is still an open problem, and there are no known methods to handle concept definitions or even (qualified) existential quantification in conjunction with the constructors that cause the finite model property not to hold. The main result of this paper, presented in Section 5, is a method to reason with respect to finite models on free $\mathcal{ALCQI}$-schemata. The possibility in such schemata to make use of full negation and of concept definitions provides indeed the necessary expressivity to go beyond the basic features of DB models and capture and reason upon several useful constructs that have been proposed as extensions to existing models. Relevant examples are:

- Classes defined through necessary and sufficient conditions, which correspond to so called *views* in object-oriented DBs (Abiteboul & Bonner, 1991). Using free assertions[5] it is also possible to specify only sufficient conditions for an object to be an instance of a class, and more general forms of integrity constraints.
- Classes defined as the union of other classes.
- ISA relations between relationships in the Entity-Relationship model (Di Battista & Lenzerini, 1993).
- Negative assertions about classes, such as disjointness, or non-participation in a relationship (Di Battista & Lenzerini, 1993).

[5]See footnote 3.

$$
\begin{array}{rcl}
\texttt{Athlete} & \dot{\preceq} & \forall \texttt{partic.Comp} \sqcap \exists^{\leq 4}\texttt{partic} \sqcap \\
 & & \forall \texttt{wins.RunningComp} \sqcap \exists \texttt{wins.Final} \\
\texttt{Comp} & \dot{\preceq} & \forall \texttt{partic}^{-}\texttt{.Athlete} \\
\texttt{Final} & \dot{\preceq} & \texttt{Comp} \sqcap \exists^{\leq 1}\texttt{wins}^{-} \\
\texttt{RunningComp} & \dot{\preceq} & \texttt{Comp} \\
\texttt{Final} \sqcap \texttt{RunningComp} & \preceq & \exists^{\geq 6}\texttt{partic}^{-}
\end{array}
$$

Figure 1: A schema with finitely inconsistent concepts

Example 3 The schema $\mathcal{S}$ shown in Figure 1 models the participation of athletes in sports competitions. The last (free) inclusion assertion represents a constraint that is not associated to a specific concept name. It turns out, however, that $\mathcal{S}$ contains a modeling error that causes `Athlete` to be finitely inconsistent. In fact, the conditions imposed on `Athlete` should only be required for (fast) runners that win a final. Therefore it is not correct to assume that only such athletes can participate in a competition. Notice that the inconsistency above would not be detected by a reasoning procedure that ignores finiteness, since all concepts in $\mathcal{S}$ are consistent. ∎

Before discussing in Section 5 the method to reason with respect to finite models, we briefly illustrate in Section 4 how to reason with respect to arbitrary models in a very expressive DL which includes the well-foundedness constructor, and therefore allows us to represent finite modeling structures.

4 REASONING ABOUT FINITE MODELING STRUCTURES

We now sketch a method to reason with respect to arbitrary models on free schemata expressed in a DL that includes all constructors shown in Table 1, and in particular the well-founded constructor that allows us to define finite modeling structures. Some of the other constructors, however, and in particular role-value maps, intersections of complex roles, and number restrictions on complex roles, make reasoning highly undecidable if they are used without restrictions (Schmidt-Schauß, 1989; Harel, 1983). Therefore, we syntactically restrict the use of these problematic constructors in a way similar to what done in (De Giacomo & Lenzerini, 1995) for the logic $\mathcal{CATS}$. The logic $\mathcal{ALT}$, in which we distinguish between *basic roles*, denoted by the letter B, and arbitrary complex roles, is defined as follows:

$$
\begin{array}{rcl}
E & \longrightarrow & C \mid E_1 \sqcap E_2 \mid \neg E \mid \forall R.E \mid \exists^{\leq n} B.E \mid \\
 & & \exists^{\leq n} B^{-}.E \mid (B_1 \subseteq B_2) \mid (B_1^{-} \subseteq B_2^{-}) \mid wf(R)
\end{array}
$$

$$B \longrightarrow P \mid B_1 \cup B_2 \mid B_1 \setminus B_2$$
$$R \longrightarrow B \mid R_1 \cup R_2 \mid R_1 \circ R_2 \mid R^- \mid R^* \mid id(E).$$

The semantics for $\mathcal{ALT}$ class and role expressions is defined as in Table 1, considering that basic roles are just particular types of complex roles.

Decidability in deterministic exponential time of reasoning in an object-oriented model with the same expressivity as free $\mathcal{ALT}$-schemata was already shown in (Calvanese, De Giacomo, & Lenzerini, 1995). We briefly sketch the idea underlying the proof, which exploits the correspondence with PDLs: Concepts of DLs correspond to formulae of PDLs, roles correspond to programs, and (almost) every constructor of DLs has its counterpart in PDLs.

We can reduce the problem of reasoning on a free $\mathcal{ALT}$-schema to the problem of verifying the satisfiability of a formula of RFCPDL, which is basic PDL augmented with the converse operator on programs (corresponding to inverse roles), local functionality on both direct and inverse atomic programs (which corresponds to functional restrictions, to which existential quantifications are reduced by "reifying" basic roles) and the repeat constructor (which corresponds to well-foundedness). The details of the reduction can be found in (Calvanese, 1996b).

Proposition 4 *Given a free $\mathcal{ALT}$-schema $\mathcal{S}$ and a concept expression E, one can construct in polynomial time in $|\mathcal{S}| + |E|$ an* RFCPDL*-formula $f_{\mathcal{S},E}$ such that $f_{\mathcal{S},E}$ is satisfiable if and only if E is consistent in $\mathcal{S}$.*

In order to prove the desired decidability result, however, we cannot exploit known decision procedures for PDLs, since the decidability of RFCPDL is open. To show decidability of RFCPDL we reduce satisfiability of a formula to nonemptiness of the language accepted by a finite automaton on infinite objects derived from the formula (Thomas, 1990). This is a standard technique that provides tight upper bounds for various modal and temporal logics (Emerson & Jutla, 1989; Vardi & Wolper, 1994) and logics of programs (Vardi, 1985; Vardi & Wolper, 1986; Vardi & Stockmeyer, 1985), and we extend it to RFCPDL. It is based on the tree model property which holds also for RFCPDL. In particular, the automaton we obtain from a RFCPDL formula f is a hybrid automaton $H_f := (T_f, W_f)$, that combines an automaton on infinite trees T_f of exponential size in $|f|$, with an automaton on infinite words W_f of polynomial size in $|f|$ and which handles the repeat sub-formulae. Exploiting the results described in (Emerson & Jutla, 1989), the nonemptiness problem for such a hybrid tree automaton can be solved in deterministic time that is polynomial in the number of states of T_f and exponential in the number of states of W_f.

This allows us to establish the desired upper bound for satisfiability in RFCPDL, and therefore for unrestricted model reasoning in $\mathcal{ALT}$. By the well known **EXPTIME** lower bound for reasoning on free $\mathcal{FL}^-$-schemata[6] the complexity bound is tight.

Theorem 5 *Unrestricted concept consistency and concept subsumption in free $\mathcal{ALT}$-schemata are **EXPTIME**-complete.*

5 REASONING WITH RESPECT TO FINITE MODELS

In this section we present a technique for reasoning with respect to finite models in $\mathcal{L}$-schemata. The method extends the one proposed in (Calvanese et al., 1994) to reason on primitive $\mathcal{ALUNI}$-schemata, and is based on the same idea of constructing from a schema a system of linear inequations, and relating the existence of finite models of the schema to the existence of particular solutions of the system. The method we present decides concept consistency (and therefore also subsumption) in free $\mathcal{ALCQI}$-schemata. Due to the presence of arbitrary free inclusion assertions, and the higher expressivity of the underlying language (in particular the presence of existential quantification) the construction of the system of inequations is much more involved than for primitive $\mathcal{ALUNI}$-schemata.

We describe the method for reasoning on free $\mathcal{ALCNI}$-schemata and sketch then how it must be extended to deal with qualified number restrictions. We use the term *literal*, ranged over by L, to denote either a concept name or a concept name preceded by the symbol "$\neg$". It is easy to see that each free schema can be transformed in linear time into an equivalent *normalized schema*, where each assertion has the form $L \doteq E$, and E has one of the forms

$$L_1 \mid L_1 \sqcup L_2 \mid \forall R.L_1 \mid \exists R.L_1 \mid \exists^{\geq m} R \mid \exists^{\leq n} R.$$

Therefore, in the following we assume to deal with free normalized $\mathcal{ALCNI}$-schemata.

5.1 EXPANSION OF A SCHEMA

We generalize now the definition of expansion of a schema introduced in (Calvanese et al., 1994) to free $\mathcal{ALCNI}$-schemata. Let $\mathcal{S} := (\mathcal{C}, \mathcal{P}, \mathcal{T})$ be such a schema. We call a set $\widehat{D} \in 2^{\mathcal{C}}$ a *compound concept*,

[6] $\mathcal{FL}^-$ is the language obtained from $\mathcal{AL}$ by dropping atomic negation.

and an element $\hat{Q} \in \mathcal{P} \times 2^{\mathcal{C}} \times 2^{\mathcal{C}}$ a *compound role*. $\hat{D}$ ranges over compound concepts and $\hat{Q}$ over compound roles. A *compound assertion* has one of the forms

$$\hat{D} \doteq \exists^{\geq m} R, \qquad \hat{D} \doteq \exists^{\leq n} R, \qquad \hat{D} \doteq \exists R.L.$$

For a negative literal $L = \neg C$, when we write $L \in \hat{D}$ we mean $C \notin \hat{D}$.

Intuitively, the instances of a compound concept $\hat{D}$ are all those objects of the domain that are instances of all concepts in $\hat{D}$ and are not instances of any concept not in $\hat{D}$. A compound role $(P, \hat{D}_1, \hat{D}_2)$ is interpreted as the restriction of role P to the pairs whose first component is an instance of $\hat{D}_1$ and whose second component is an instance of $\hat{D}_2$. More formally, the semantics of compound concepts and roles is given by extending the interpretation function as follows:

$$\begin{aligned} \hat{D}^{\mathcal{I}} &:= \bigcap_{C \in \hat{D}} C^{\mathcal{I}} \setminus \bigcup_{C \in (\mathcal{C} \setminus \hat{D})} C^{\mathcal{I}} \\ (P, \hat{D}_1, \hat{D}_2)^{\mathcal{I}} &:= P^{\mathcal{I}} \cap (\hat{D}_1^{\mathcal{I}} \times \hat{D}_2^{\mathcal{I}}). \end{aligned}$$

Note that according to this definition two different compound concepts have necessarily disjoint interpretations.

We say that a compound concept $\hat{D} \in 2^{\mathcal{C}}$ is *$\mathcal{S}$-consistent*, if for every $L \in \hat{D}$,

- if $(L \doteq L') \in \mathcal{T}$, then $L' \in \hat{D}$, and
- if $(L \doteq L_1 \sqcup L_2) \in \mathcal{T}$, then $L_1 \in \hat{D}$ or $L_2 \in \hat{D}$.

We say that a compound role $(P, \hat{D}_1, \hat{D}_2) \in \mathcal{P} \times 2^{\mathcal{C}} \times 2^{\mathcal{C}}$ is *$\mathcal{S}$-consistent*, if

- $\hat{D}_1$ and $\hat{D}_2$ are $\mathcal{S}$-consistent,
- for every $L \in \hat{D}_1$, if $(L \doteq \forall P.L') \in \mathcal{T}$, then $L' \in \hat{D}_2$, and
- for every $L \in \hat{D}_2$, if $(L \doteq \forall P^{-1}.L') \in \mathcal{T}$, then $L' \in \hat{D}_1$.

Intuitively, compound concepts and roles that are not consistent necessarily have an empty extension in all models of the schema.

The *expansion* $\hat{\mathcal{S}} := (\mathcal{C}, \mathcal{P}, \hat{\mathcal{D}}, \hat{\mathcal{Q}}, \hat{\mathcal{T}})$ of $\mathcal{S}$ is defined as follows:

- $\hat{\mathcal{D}}$ is the set of all $\mathcal{S}$-consistent compound concepts.
- $\hat{\mathcal{Q}}$ is the set of all $\mathcal{S}$-consistent compound roles.
- $\hat{\mathcal{T}}$ is the smallest set of compound assertions such that for every $\hat{D} \in \hat{\mathcal{D}}$:
  - if for some $L \in \hat{D}$ there is an assertion $(L \doteq \exists^{\geq m} R) \in \mathcal{T}$, then $\hat{\mathcal{T}}$ contains the compound assertion $\hat{D} \doteq \exists^{\geq m_{max}} R$, where $m_{max} := \max\{m \mid \exists L \in \hat{D} \colon (L \doteq \exists^{\geq m} R) \in \mathcal{T}\}$;
  - if for some $L \in \hat{D}$ there is an assertion $(L \doteq \exists^{\leq n} R) \in \mathcal{T}$, then $\hat{\mathcal{T}}$ contains the compound assertion $\hat{D} \doteq \exists^{\leq n_{min}} R$, where $n_{min} := \min\{n \mid \exists L \in \hat{D} \colon (L \doteq \exists^{\leq n} R) \in \mathcal{T}\}$.
  - if for some $L \in \hat{D}$ there is an assertion $(L \doteq \exists R.L') \in \mathcal{T}$, then $\hat{\mathcal{T}}$ contains the compound assertion $\hat{D} \doteq \exists R.L'$.

It is easy to see that the size of the expansion is exponential in the size of the original schema and that it can also be effectively constructed in exponential time.

In (Calvanese et al., 1994) it is shown that for primitive $\mathcal{ALUNI}$-schemata a system of linear inequations can be directly constructed from the compound assertions in the expansion, such that particular solutions of this system correspond to finite models of the schema. Unfortunately, for free $\mathcal{ALCNI}$-schemata this approach does not work directly, and this is due to the presence of assertions of the form $L \doteq \exists R.L'$. One could think to handle such assertions by constructing the expansion as specified above, simply coding existential quantifications inside compound assertions and leaving their treatment to the system of inequations (as done for number restrictions).

The intuitive reason why this simple approach does not lead to the desired results, is that it relies on the uniformity of all objects that are instances of the same compound concept. When setting up the system of inequations we are in fact transforming local constraints on the number of connections that a single object may have into global constraints on the total number of connections of a certain type. The necessary differentiation is introduced by constructing the expansion. Once this is done all instances of the same compound concept can be regarded as equivalent. The approach works for the case of $\mathcal{ALUNI}$-schemata, where the concept expressions cannot distinguish between different instances of the same compound concept. If we use existential quantification, however, due to the increased expressivity it is not sufficient anymore to split the schema into compound concepts to obtain a uniform behaviour of the instances. This leads us to introduce the notion of biexpansion of a schema, where we make a more fine-grained separation based also on the existence of connections of certain types.

5.2 BIEXPANSION OF A SCHEMA

In order to define the biexpansion of a schema we need some additional terminology. Let $\mathcal{P}^- := \{P^- \mid P \in \mathcal{P}\}$ and $\mathcal{R} := \mathcal{P} \cup \mathcal{P}^-$. We use $\mathcal{B}_{\mathcal{C},\mathcal{P}}$ to denote $2^{\mathcal{C}} \times (2^{2^{\mathcal{P}}})^{\mathcal{R}}$. An element of $\mathcal{B}_{\mathcal{C},\mathcal{P}}$ is called a *bicompound concept*, and it is constituted by a pair $(\hat{D}, \varphi)$ in

which the first element is a compound concept, and the second element is a function $\varphi : \mathcal{R} \to 2^{2^{\mathcal{C}}}$ that associates to each role and inverse role a set of compound concepts. We introduce two functions that allow us to refer to the components of bicompound concepts:

- The function cc : $\mathcal{B}_{\mathcal{C},\mathcal{P}} \to 2^{\mathcal{C}}$ returns the first component $\widehat{D}$ of a bicompound concept $(\widehat{D}, \varphi)$.
- The function ccs : $\mathcal{B}_{\mathcal{C},\mathcal{P}} \times \mathcal{R} \to 2^{2^{\mathcal{C}}}$ returns for a bicompound concept $\widetilde{B} := (\widehat{D}, \varphi)$ and a role R the set of compound concepts assigned to R by the second component φ of $\widetilde{B}$.

An element of $\mathcal{P} \times \mathcal{B}_{\mathcal{C},\mathcal{P}} \times \mathcal{B}_{\mathcal{C},\mathcal{P}}$ is called a *bicompound role*. $\widetilde{B}$ and $\widetilde{U}$ range over bicompound concepts and bicompound roles respectively.

The intuition behind the definition of bicompound concepts and roles is the following: If $\mathrm{cc}(\widetilde{B}) = \widehat{D}$, then all instances of $\widetilde{B}$ are instances of $\widehat{D}$. Let o be such an instance. Then for each compound concept $\widehat{D}'$ in $\mathrm{ccs}(\widetilde{B}, R)$, there is an instance o' of $\widehat{D}'$ connected to o via role R, while for the compound concepts not in $\mathrm{ccs}(\widetilde{B}, R)$ there is no such instance. In analogy to compound roles, a bicompound role $(P, \widetilde{B}_1, \widetilde{B}_2)$ is interpreted as the restriction of role P to the pairs whose first component is an instance of $\widetilde{B}_1$ and whose second component is an instance of $\widetilde{B}_2$.

A *bicompound assertion* has one of the forms

$$\widetilde{B} \doteq \exists^{\geq m} R, \qquad \widetilde{B} \doteq \exists^{\leq n} R.$$

The semantics of bicompound concepts and roles is again defined by extending the interpretation function, where for simplicity of notation we make use of concept expressions built by applying the $\mathcal{ALCNI}$-constructors to both concepts and compound concepts:

$$\widetilde{B}^{\mathcal{I}} := (\mathrm{cc}(\widetilde{B}))^{\mathcal{I}} \cap \bigcap_{R \in \mathcal{R}} \left(\bigcap_{\widehat{D} \in \mathrm{ccs}(\widetilde{B},R)} (\exists R.\widehat{D})^{\mathcal{I}} \cap \bigcap_{\widehat{D} \in 2^{\mathcal{C}} \setminus \mathrm{ccs}(\widetilde{B},R)} (\neg \exists R.\widehat{D})^{\mathcal{I}} \right)$$

$$(P, \widetilde{B}_1, \widetilde{B}_2)^{\mathcal{I}} := P^{\mathcal{I}} \cap (\widetilde{B}_1^{\mathcal{I}} \times \widetilde{B}_2^{\mathcal{I}})$$

This definition, together with the fact that different compound concepts have disjoint extensions, implies that two different bicompound concepts also have disjoint extensions. The same observation holds for two different bicompound roles $(P, \widetilde{B}_1, \widetilde{B}_2)$ and $(P, \widetilde{B}_1', \widetilde{B}_2')$ that refer to the same role P. Also, given the extensions of all bicompound concepts and bicompound roles, it is immediate to obtain the extensions of concepts and roles.

We say that a bicompound concept $\widetilde{B} \in \mathcal{B}_{\mathcal{C},\mathcal{P}}$ is $\mathcal{S}$-consistent, if

- $\mathrm{cc}(\widetilde{B})$ is $\mathcal{S}$-consistent,
- for every $R \in \mathcal{R}$, for every $\widehat{D} \in \mathrm{ccs}(\widetilde{B}, R)$, $\widehat{D}$ is $\mathcal{S}$-consistent,
- for every $R \in \mathcal{R}$, for every $L \in \mathrm{cc}(\widetilde{B})$, if $(L \doteq \exists R.L') \in \mathcal{T}$, then there is a $\widehat{D} \in \mathrm{ccs}(\widetilde{B}, R)$ such that $L' \in \widehat{D}$, and
- for every $R \in \mathcal{R}$, for every $L \in \mathrm{cc}(\widetilde{B})$, if $(L \doteq \forall R.L') \in \mathcal{T}$, then for all $\widehat{D} \in \mathrm{ccs}(\widetilde{B}, R)$ it holds that $L' \in \widehat{D}$.

We say that a bicompound role $(P, \widetilde{B}_1, \widetilde{B}_2) \in \mathcal{P} \times \mathcal{B}_{\mathcal{C},\mathcal{P}} \times \mathcal{B}_{\mathcal{C},\mathcal{P}}$ is $\mathcal{S}$-consistent, if

- $\widetilde{B}_1$ and $\widetilde{B}_2$ are $\mathcal{S}$-consistent, $\mathrm{cc}(\widetilde{B}_2) \in \mathrm{ccs}(\widetilde{B}_1, P)$, $\mathrm{cc}(\widetilde{B}_1) \in \mathrm{ccs}(\widetilde{B}_2, P^-)$,
- for every $L \in \mathrm{cc}(\widetilde{B}_1)$, if $(L \doteq \forall P.L') \in \mathcal{T}$, then $L' \in \mathrm{cc}(\widetilde{B}_2)$, and
- for every $L \in \mathrm{cc}(\widetilde{B}_2)$, if $(L \doteq \forall P^-.L') \in \mathcal{T}$, then $L' \in \mathrm{cc}(\widetilde{B}_1)$.

Again, $\mathcal{S}$-consistency reflects the assertions that constitute the schema, and only $\mathcal{S}$-consistent bicompound concepts and roles can be populated in some model of $\mathcal{S}$.

The *biexpansion* $\widetilde{\mathcal{S}} = (\mathcal{C}, \mathcal{P}, \widetilde{\mathcal{B}}, \widetilde{\mathcal{U}}, \widetilde{\mathcal{T}})$ of $\mathcal{S}$ is defined as follows:

- $\widetilde{\mathcal{B}}$ is the set of all $\mathcal{S}$-*consistent* bicompound concepts.
- $\widetilde{\mathcal{U}}$ is the set of all $\mathcal{S}$-*consistent* bicompound roles.
- $\widetilde{\mathcal{T}}$ is the smallest set of bicompound assertions such that for every $\widetilde{B} \in \widetilde{\mathcal{B}}$:
  - if for some $L \in \mathrm{cc}(\widetilde{B})$ there is an assertion $(L \doteq \exists^{\geq m} R) \in \mathcal{T}$, then $\widetilde{\mathcal{T}}$ contains the bicompound assertion $\widetilde{B} \doteq \exists^{\geq m_{max}} R$, where $m_{max} := \max\{m \mid \exists L \in \mathrm{cc}(\widetilde{B}) : (L \doteq \exists^{\geq m} R) \in \mathcal{T}\}$.
  - if for some $L \in \mathrm{cc}(\widetilde{B})$ there is an assertion $(L \doteq \exists^{\leq n} R) \in \mathcal{T}$, then $\widetilde{\mathcal{T}}$ contains the bicompound assertion $\widetilde{B} \doteq \exists^{\leq n_{min}} R$, where $n_{min} := \min\{n \mid \exists L \in \mathrm{cc}(\widetilde{B}) : (L \doteq \exists^{\leq n} R) \in \mathcal{T}\}$.

It is easy to see that the size of the biexpansion is in general double exponential in the size of the original schema, and it can also be effectively constructed in double exponential time.

5.3 CHARACTERIZATION OF FINITE MODEL REASONING

We are now ready to define a system $\Psi_{\mathcal{S}}$ of linear inequations whose solutions of a certain type are related to the finite models of a schema $\mathcal{S}$. Let $\widetilde{\mathcal{S}} =$

$(\mathcal{C}, \mathcal{P}, \widetilde{\mathcal{B}}, \widetilde{\mathcal{U}}, \widetilde{\mathcal{T}})$ be the biexpansion of $\mathcal{S}$. Then $\Psi_{\mathcal{S}}$ contains one unknown $\mathrm{Var}(\widetilde{X})$ for each bicompound concept or role $\widetilde{X}$ in $\widetilde{\mathcal{B}} \cup \widetilde{\mathcal{U}}$, and consists of the following linear homogeneous inequations:

- For each $\widetilde{B} \in \widetilde{\mathcal{B}}$, for each $P \in \mathcal{P}$, for each $\widehat{D} \in \mathrm{ccs}(\widetilde{B}, P)$ the inequation

$$\mathrm{Var}(\widetilde{B}) \leq \sum_{(P,\widetilde{B},\widetilde{B}_2) \in \widetilde{\mathcal{U}} \mid \mathrm{cc}(\widetilde{B}_2) = \widehat{D}} \mathrm{Var}((P, \widetilde{B}, \widetilde{B}_2)).$$

- For each $\widetilde{B} \in \widetilde{\mathcal{B}}$, for each $P \in \mathcal{P}$, for each $\widehat{D} \in \mathrm{ccs}(\widetilde{B}, P^-)$ the inequation

$$\mathrm{Var}(\widetilde{B}) \leq \sum_{(P,\widetilde{B}_1,\widetilde{B}) \in \widetilde{\mathcal{U}} \mid \mathrm{cc}(\widetilde{B}_1) = \widehat{D}} \mathrm{Var}((P, \widetilde{B}_1, \widetilde{B})).$$

- For each bicompound assertion $(\widetilde{B} \doteq \exists^{\geq m} R) \in \widetilde{\mathcal{T}}$ the inequation

$$m \cdot \mathrm{Var}(\widetilde{B}) \leq S(\widetilde{B}, R), \qquad \text{where}$$

$$S(\widetilde{B}, P) := \sum_{(P,\widetilde{B},\widetilde{B}_2) \in \widetilde{\mathcal{U}}} \mathrm{Var}((P, \widetilde{B}, \widetilde{B}_2)),$$

$$S(\widetilde{B}, P^-) := \sum_{(P,\widetilde{B}_1,\widetilde{B}) \in \widetilde{\mathcal{U}}} \mathrm{Var}((P, \widetilde{B}_1, \widetilde{B})).$$

- For each bicompound assertion $(\widetilde{B} \doteq \exists^{\leq n} R) \in \widetilde{\mathcal{T}}$ the inequation

$$n \cdot \mathrm{Var}(\widetilde{B}) \geq S(\widetilde{B}, R).$$

It is now possible to relate a solution of $\Psi_{\mathcal{S}}$ to a finite model of $\mathcal{S}$, in which each bicompound concept and bicompound role has a number of instances that is given by the value that the solution assigns to the corresponding unknown. Since in a model of $\mathcal{S}$ each bicompound role that refers (in its second or third component) to a bicompound concept that has an empty extension, also has an empty extension, this condition reflects in an additional requirement that the solution must satisfy. A solution of $\Psi_{\mathcal{S}}$ is *acceptable*, if, whenever it assigns value 0 to an unknown corresponding to a bicompound concept $\widetilde{B}$, then it assigns also value 0 to all unknowns corresponding to bicompound roles that have $\widetilde{B}$ as their second or third component. We can now state the main theorem concerning finite concept consistency.

Theorem 6 *A concept $C \in \mathcal{C}$ is finitely consistent in $\mathcal{S}$ if and only if*

$$\Psi_{\mathcal{S}}^{C} := \Psi_{\mathcal{S}} \bigcup \left\{ \sum_{\widetilde{B} \in \widetilde{\mathcal{B}} \mid C \in \mathrm{cc}(\widetilde{B})} Var(\widetilde{B}) \geq 1 \right\}$$

admits an acceptable nonnegative integer solution.

As shown in (Calvanese, 1996b), by applying linear programming techniques one can decide the existence of acceptable nonnegative integer solutions in polynomial time in the size of the system. Since $\Psi_{\mathcal{S}}$ can be constructed in time which is at most double exponential in $|\mathcal{S}|$, and since in free $\mathcal{ALCNI}$-schemata we can easily reduce both consistency of an arbitrary concept expression and concept subsumption to consistency of a single concept name, we obtain the following worst case upper bound.

Theorem 7 *Finite concept consistency and concept subsumption in free $\mathcal{ALCNI}$-schemata can be decided in worst case deterministic double exponential time.*

The method as described above cannot deal directly with qualified number restrictions, since the differentiation introduced by bicompound concepts is too coarse. In fact, we need to make a separation in bicompound classes based not only on the existence but also on the number of links of a certain type. It turns out that it is in fact sufficient to consider only intervals of numbers of links, where the ranges of these intervals are given by the numbers that effectively appear in the schema. In this way it is still possible to keep the size of the resulting "extended" biexpansion double exponential in the size of the schema. The resulting "extended" bicompound assertions can then again be coded by means of a system of linear inequations, and we can prove the counterpart of Theorem 6 for the system derived from a free $\mathcal{ALCQI}$-schema. We omit further details and state only the final result.

Theorem 8 *Finite concept consistency and concept subsumption in free $\mathcal{ALCQI}$-schemata can be decided in worst case deterministic double exponential time.*

6 CONCLUSIONS

In this paper we have discussed the issues concerning finite model reasoning at the intensional level in expressive DLs that lack the finite model property. Two distinct ways of dealing with finiteness have been proposed:

1. We have introduced in the language a specific constructor to express well-foundedness, and have shown how known methods for reasoning with respect to arbitrary models can be extended to deal with it. This provides an **EXPTIME** decision procedure for concept consistency and subsumption.

2. We have developed a technique to reason with respect to finite models on schemata of the most general form expressed in a DL capable of capturing most known known data models. The proposed algorithm works in worst case deterministic double exponential time.

We are currently extending our work in several directions. A major point is the extension of the reasoning techniques to deal also with extensional knowledge, i.e. with assertions stating properties of specific objects. In the unrestricted case, the reasoning technique for $\mathcal{ALT}$ discussed in Section 4 can be directly integrated with the work in (De Giacomo, 1996), still obtaining an **EXPTIME** upper-bound. In the finite case, the proposed technique can be applied under specific assumptions which are rather restrictive but common in DBs. However, a general method for finite model reasoning in expressive DLs on a schema together with assertions on individuals is still open.

We aim also at increasing the expressivity of the schema definition language while still preserving decidability. Concerning point 1 above, we have mentioned in Section 3 that the well-founded constructor is equivalent to a particular type of least-fixpoint expression. A natural extension would be to allow arbitrary nested fixpoints as in the μ-calculus (Kozen, 1983) together with inverse roles and number restrictions. Concerning point 2, it can be shown that the reasoning method developed for $\mathcal{ALCQI}$ can be extended to deal with some of the constructors of $\mathcal{ALT}$, and in particular basic roles (i.e. union, intersection, and difference of role names and inverse roles) and role value maps on basic roles. Qualified number restrictions turn out to be indispensable for this. If the addition of reflexive transitive closure preserves decidability also in the finite case is still open.

Acknowledgements

I would like to thank Maurizio Lenzerini and Giuseppe De Giacomo for many inspiring discussions and Franz Baader for his hospitality at the RWTH Aachen, where part of this work was carried out.

References

Abiteboul, S. & Bonner, A. (1991). Objects and views. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pp. 238–247 New York (NY, USA).

Artale, A., Franconi, E., Guarino, N., & Pazzi, L. (1996). Part-whole relations in object-centered systems: An overview. *Data and Knowledge Engineering*. To appear.

Baader, F. (1990). Terminological cycles in KL-ONE-based knowledge representation languages. In *Proc. of the 8th Nat. Conf. on Artificial Intelligence (AAAI-90)*, pp. 621–626 Boston, Ma.

Bergamaschi, S. & Sartori, C. (1992). On taxonomic reasoning in conceptual design. *ACM Transactions on Database Systems*, *17*(3), 385–422.

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

Borgida, A., Lenzerini, M., Nardi, D., & Nebel, B. (Eds.). (1995). *Working Notes of the 1995 Description Logics Workshop*, Technical Report, RAP 07.95, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Rome (Italy).

Buchheit, M., Donini, F. M., Nutt, W., & Schaerf, A. (1995). A refined architecture for terminological systems: Terminology = schema + views. Tech. rep. RR-95-09, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI).

Buongarzoni, P., Menghini, C., Salis, R., Sebastiani, F., & Straccia, U. (1995). Logical and computational properties of the description logic MIRTL.. In Borgida et al. (Borgida, Lenzerini, Nardi, & Nebel, 1995), pp. 80–84.

Calvanese, D. (1996a). Reasoning with inclusion axioms in description logics: Algorithms and complexity. In *Proc. of the 12th European Conf. on Artificial Intelligence (ECAI-96)*, pp. 303–307. John Wiley & Sons.

Calvanese, D. (1996b). *Unrestricted and Finite Model Reasoning in Class-Based Representation Formalisms.* Ph.D. thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza".

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 LNCS, pp. 229–246. Springer-Verlag.

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

Cattell, R. G. G. (Ed.). (1994). *The Object Database Standard: ODMG-93.* Morgan Kaufmann. Release 1.1.

De Giacomo, G. (1996). Tbox and abox reasoning in expressive description logics. In *Proc. of the 5th Int.*

Conf. on the Principles of Knowledge Representation and Reasoning (KR-96). Morgan Kaufmann.

De Giacomo, G. & Lenzerini, M. (1994a). Boosting the correspondence between description logics and propositional dynamic logics. In *Proc. of the 12th Nat. Conf. on Artificial Intelligence (AAAI-94)*, pp. 205–212. AAAI Press/The MIT Press.

De Giacomo, G. & Lenzerini, M. (1994b). Concept language with number restrictions and fixpoints, and its relationship with μ-calculus. In *Proc. of the 11th European Conf. on Artificial Intelligence (ECAI-94)*, pp. 411–415.

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.

Di Battista, G. & Lenzerini, M. (1993). Deductive entity-relationship modeling. *IEEE Transactions on Knowledge and Data Engineering*, *5*(3), 439–450.

Donini, F. M., Lenzerini, M., Nardi, D., & Nutt, W. (1991a). The complexity of concept languages. In *Proc. of the 2nd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-91)*, pp. 151–162. Morgan Kaufmann.

Donini, F. M., Lenzerini, M., Nardi, D., & Nutt, W. (1991b). Tractable concept languages. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI-91)*, pp. 458–463 Sydney.

Emerson, E. A. & Jutla, C. S. (1989). On simultaneously determinizing and complementing ω-automata. In *Proc. of the 4th Int. Conf. of Logic in Computer Science (LICS-89)*, pp. 333–342.

Harel, D. (1983). Recurring dominoes: Making the highly undecidable highly understandable. In *Proc. of the Int. Conf. on Foundations of Computational Theory (FCT-83)*, No. 158 in LNCS, pp. 177–194. Springer-Verlag.

Kozen, D. (1983). Results on the propositional μ-calculus. *Theoretical Computer Science*, *27*, 333–354.

Kozen, D. & Tiuryn, J. (1990). Logics of programs. In *Handbook of Theoretical Computer Science - Formal Models and Semantics*, pp. 789–840. Elsevier Science Publishers (North-Holland).

Lecluse, C. & Richard, P. (1989). Modeling complex structures in object-oriented databases. In *Proc. of the 8th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS-89)*, pp. 362–369.

Nebel, B. (1990). Terminological reasoning is inherently intractable. *Artificial Intelligence Journal*, *43*, 235–249.

Nebel, B. (1991). Terminological cycles: Semantics and computational properties. In Sowa, J. F. (Ed.), *Principles of Semantic Networks*, pp. 331–361. Morgan Kaufmann.

Sattler, U. (1995). A concept language for an engineering application with part-whole relations.. In Borgida et al. (Borgida et al., 1995), pp. 119–123.

Schild, K. (1991). A correspondence theory for terminological logics: Preliminary report. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI-91)*, pp. 466–471 Sydney, Australia.

Schild, K. (1994). Terminological cycles and the propositional μ-calculus. In *Proc. of the 4th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-94)*, pp. 509–520 Bonn. Morgan Kaufmann.

Schmidt-Schauß, M. (1989). Subsumption in KL-ONE is undecidable. In *Proc. of the 1st Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-89)*, pp. 421–431. Morgan Kaufmann.

Streett, R. E. (1982). Propositional dynamic logic of looping and converse is elementarily decidable. *Information and Computation*, *54*, 121–141.

Thomas, W. (1990). Automata on infinite objects. In van Leuven, J. (Ed.), *Handbook of Theoretical Computer Science*, Vol. B, chap. 4, pp. 134–189. Elsevier Science Publishers (North-Holland).

Vardi, M. Y. (1985). The taming of converse: Reasoning about two-way computations. In *Proc. of the 4th Workshop on Logics of Programs*, No. 193 in LNCS, pp. 413–424. Springer-Verlag.

Vardi, M. Y. & Stockmeyer, L. (1985). Improved upper and lower bounds for modal logics of programs: Preliminary report. In *Proc. of the 17th ACM SIGACT Sym. on Theory of Computing (STOC-85)*, pp. 240–251.

Vardi, M. Y. & Wolper, P. (1986). Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Sciences*, *32*, 183–221. A preliminary version appeared in *Proc. of the 16th ACM SIGACT Symp. on Theory of Computing (STOC-84)*.

Vardi, M. Y. & Wolper, P. (1994). Reasoning about infinite computations. *Information and Computation*, *115*(1), 1–37.

A *SAT-based* decision procedure for $\mathcal{ALC}$

Fausto Giunchiglia
IRST, 38050 Povo, Trento, Italy.
DISA, Università di Trento, Italy.
fausto@irst.itc.it

Roberto Sebastiani
DIST, Università di Genova,
v. Causa 15, 16146 Genova, Italy.
rseba@mrg.dist.unige.it

Abstract

The goal of this paper is to describe and thoroughly test a decision procedure, called KSAT, checking satisfiability in the terminological logic $\mathcal{ALC}$. KSAT is said to be *SAT-based* as it is defined in terms of a decision procedure for propositional satisfiability (SAT). The tests are performed comparing KSAT with, among other procedures, KRIS, a state-of-the-art tableau-based implementation of a decision procedure for $\mathcal{ALC}$. KSAT outperforms KRIS of orders of magnitude. Furthermore, the empirical results highlight an intrinsic weakeness that tableau-based decision procedures have with respect to SAT-based decision procedures.

1 INTRODUCTION

The goal of this paper is to describe and thoroughly test a new decision procedure, called KSAT, checking satisfiability in the terminological logic $\mathcal{ALC}$, as defined in (Schmidt-Schauß & Smolka 1991), comprising Boolean operations on concepts and value restrictions, and not restricted to CNF form.[1]

As it is well known, $\mathcal{ALC}$ is a notational variant of K(m), that is, K with m modalities (Schild 1991).[2] The main idea underlying the definition of KSAT is that a decision procedure for satisfiability in K(m) (K(m)-satisfiability) can be defined in terms of a decision procedure for propositional satisfiability (SAT). As a matter of terminology, we call *SAT-based* all the decision procedures whose definition is based on this idea.[3]

KSAT outperforms all the decision procedures and systems for terminological and modal logics we have been able to acquire. In this paper we compare KSAT with two of them. The first is a tableau-based procedure — due to B. Nebel and E. Franconi — which is essentially a straightforward implementation of the algorithm described in (Hollunder, Nutt & Schmidt-Schauß 1990). This procedure is called TABLEAU from now on. [4] The second is the state-of-the-art system KRIS described in (Hollunder et al. 1990, Baader, Franconi, Hollunder, Nebel & Profitlich 1994). [5] There are many reasons why a system can be more efficient than another. A crucial one is the "smartness" of the implementation. The implementation of KSAT we use is naive in many respects, e.g., it is in Lisp and it does not use fancy optimized data structures. We still have to push our work in this direction. KSAT is smarter than its competitors for a much more interesting reason. Both TABLEAU and KRIS are tableau-based. As Section 4 describes in detail, tableau-based decision procedures have an intrinsic weakness which makes it very hard if not impossible to be as efficient as SAT-based decision procedures. In our opinion, this is the most interesting theoretical result of this paper.

[1] KSAT, the test code and all the results presented in this paper are available via anonymous FTP at ftp.mrg.dist.unige.it in the directory pub/mrg-systems/ksat/ksat1.

[2] In this paper we always refer to K(m) rather than to $\mathcal{ALC}$. In particular, we speak of wffs rather than concepts, modalities rather than roles, and so on. K(m)'s syntax is simpler than $\mathcal{ALC}$'s. Notice however that the current implementation of KSAT works with $\mathcal{ALC}$'s syntax.

[3] Although this is beyond the goals of this paper, it is worth noticing that this methodology is general, and can be extended to the other normal and (we think) non normal logics, following the methodology and results presented in (Giunchiglia & Serafini 1994, Giunchiglia, Serafini, Giunchiglia & Frixione 1993) (but see also, e.g., (Fitting 1983, Massacci 1994)).

[4] TABLEAU is available via anonymous FTP at ftp.mrg.dist.unige.it in pub/mrg-systems/tableau.

[5] KRIS is available via anonymous FTP at ftp.dfki.uni-sb.de in /pub/tacos/KRIS.

The paper is structured as follows. In Section 2 we present the algorithm implemented by KSAT. In Section 3 we briefly survey our test methodology, originally defined in (Giunchiglia & Sebastiani 1996). This material is needed for a correct understanding of the experimental results reported later. In Section 4 we perform a comparative analysis of a first set of experimental results. This analysis allows us to show why SAT-based decision procedures are intrinsecally more efficient than tableau-based decision procedures. Finally, in Section 5 we perform an exhaustive empirical analysis of KSAT and KRIS, that is, the fastest SAT-based and the fastest tableau-based decision procedure at our disposal. This allows us to confirm the analysis done in Section 4 and, looking at the KSAT results, to get a better understanding of where the hardest cases are. Among other things, this allows us to reveal what looks like a phase transition (Mitchell, Selman & Levesque 1992, Williams & Hogg 1994). To our knowledge this is the first time that this phenomenon has been found in a modal logic.

The analysis presented in this paper builds on and takes to its conclusion the work preliminarily described in (Giunchiglia & Sebastiani 1996). It improves on the previous material in three important aspects. Let us call KSAT$_0$ the decision procedure presented in (Giunchiglia & Sebastiani 1996) (called KSAT in (Giunchiglia & Sebastiani 1996)). [6] First, the algorithm and its heuristics, are extended from dealing with a single modality to dealing with multiple modalities. Second, the implementation is improved. KSAT is much faster than KSAT$_0$ (in our tests, up to two orders of magnitude, see Sections 4 and 5). This has been obtained essentially by adding an initial phase of wff preprocessing. Other — relatively minor — implementational variations can be understood by comparing the code of the two systems. Third, and more important, the testing in (Giunchiglia & Sebastiani 1996) was not exhaustive and only compared KSAT$_0$ with TABLEAU. This made us miss some important points, and the phenomena described in Sections 4 and 5 went unnoticed. Furthermore, the increased efficiency of KSAT$_0$ with respect to TABLEAU in (Giunchiglia & Sebastiani 1996) was wrongly motivated by the efficiency of the propositional decision procedure. KSAT and KSAT$_0$ (when applied to a single modality) implement essentially the same algorithm. KSAT is only a more efficient implementation. The same applies to KRIS and TABLEAU. As the results in Section 5 show, the move from KSAT$_0$ to KSAT, or from TABLEAU to KRIS causes an increase in efficiency, but it does not change the shape of the efficiency curves, as it happens in the move from TABLEAU to KSAT$_0$ (or from KRIS to KSAT).

2 THE ALGORITHM

Let us write $\Box_r$ to mean the r-th modality. Let us call *atom* any wff which can not be decomposed propositionally, and *modal atom* any atom of the form $\Box_r\psi$. Let φ be the modal wff to be proved satisfiable. The algorithm for testing K(m)-satisfiability follows two basic steps, the first implementing the propositional reasoning, the second implementing the modal reasoning:

1. Using a decision procedure for propositional satisfiability, assign a truth value to (a subset of) the atoms occurring in φ in a way to make φ evaluate to T. Let us call *truth assignment (for φ)* the resulting set μ of truth value assignments. We say that μ *propositionally satisfies* φ. [7] Then μ is of the form

$$\begin{aligned}\mu \;= \{&\Box_1\alpha_{11} = T, \Box_1\alpha_{12} = T, \ldots,\\ &\Box_1\beta_{11} = F, \Box_1\beta_{12} = F, \ldots,\\ &\ldots\\ &\Box_m\alpha_{m1} = T, \Box_m\alpha_{m2} = T, \ldots,\\ &\Box_m\beta_{m1} = F, \Box_m\beta_{m2} = F, \ldots,\\ &A_1 = T, A_2 = T, \ldots,\\ &A_{R+1} = F, A_{R+2} = F, \ldots\}\end{aligned}$$

Notationally, from now on we write μ as

$$\begin{aligned}\mu \;= \;&\bigwedge_i \Box_1\alpha_{1i} \wedge \bigwedge_j \neg\Box_1\beta_{1j} \wedge\\ &\ldots\\ &\bigwedge_i \Box_m\alpha_{mi} \wedge \bigwedge_j \neg\Box_m\beta_{mj} \wedge \gamma\end{aligned} \qquad (1)$$

where $\gamma = \bigwedge_{k=1}^{R} A_k \wedge \bigwedge_{h=R+1}^{S} \neg A_h$ is a conjunction of propositional literals. Furthermore we use the greek letters μ, η to represent truth assignments.

2. Prove that the input wff φ is K(m)-satisfiable by finding (among all the possible truth assignments) a K(m)-satisfiable truth assignment μ of form as in (1). μ is K(m)-satisfiable iff the restricted assignment

$$\mu^r = \bigwedge_i \Box_r\alpha_{ri} \wedge \bigwedge_j \neg\Box_r\beta_{rj} \qquad (2)$$

[6] KSAT$_0$, the test code and all the results presented in (Giunchiglia & Sebastiani 1996) are available via anonymous FTP at `ftp.mrg.dist.unige.it` in `pub/mrg-systems/ksat/ksat0`.

[7] Notice that it is not necessary for a truth assignment to assign *all* the atoms of φ. For instance, $\{\Box_1\psi_1 = T\}$ propositionally satisfies $\Box_1\psi_1 \vee \Box_2\psi_2$.

```
function KSAT(φ)
    return KSAT_W(φ, T);

function KSAT_W(φ, μ)
    if φ = T                                          /* base      */
        then return KSAT_A(μ);
    if φ = F                                          /* backtrack */
        then return False;
    if {a unit clause (l) occurs in φ}                /* unit      */
        then return KSAT_W(assign(l, φ), μ ∧ l);
    l := choose-literal(φ);                           /* split     */
    return  KSAT_W(assign(l, φ), μ ∧ l)  or
            KSAT_W(assign(¬l, φ), μ ∧ ¬l);

function KSAT_A(⋀_i □_1 α_1i ∧ ⋀_j ¬□_1 β_1j ∧ ... ∧ ⋀_i □_m α_mi ∧ ⋀_j ¬□_m β_mj ∧ γ)
    for any box index r do
        if not KSAT_RA(⋀_i □_r α_ri ∧ ⋀_j ¬□_r β_rj)
            then return False;
    return True;

function KSAT_RA(⋀_i □_r α_ri ∧ ⋀_j ¬□_r β_rj)
    for any conjunct "¬□_r β_rj" do
        if not KSAT(⋀_i α_ri ∧ ¬β_rj)
            then return False;
    return True;
```

Figure 1: The basic version of KSAT algorithm.

is K(m)-satisfiable, for every r. μ^r is K(m)-satisfiable iff the wff

$$\varphi^{rj} = \bigwedge_i \alpha_{ri} \wedge \neg\beta_{rj} \tag{3}$$

is K(m)-satisfiable, for every j. If no truth assignment is found which is K(m)-satisfiable, then φ is not K(m)-satisfiable (K(m)-unsatisfiable).

The two steps recurse until we get to a truth assignment with no modal atoms.

The algorithm is implemented by the function KSAT in Figure 1. KSAT takes in input a modal propositional wff φ and returns a truth value asserting whether φ is K(m)-satisfiable or not. KSAT invokes directly KSAT_W (where "$_W$" stands for "*W*ff"), passing as arguments φ and the truth value T (i.e., by (1), the empty truth assignment). KSAT_W tries to build a K(m)-satisfiable truth assignment μ satisfying φ. This is done recursively, according to the following steps:

- (base) If $\varphi = T$, then μ satisfies φ. Thus, if μ is K(m)-satisfiable, then φ is K(m)-satisfiable. Therefore KSAT_W invokes $\text{KSAT}_A(\mu)$ (where "$_A$" stands for (truth) *A*ssignment). KSAT_A returns a truth value asserting whether μ is K(m)-satisfiable or not.
- (backtrack) If $\varphi = F$, then μ can not be a truth assignment for φ. Therefore KSAT_W returns *False*.
- (unit) If a literal l occurs in φ as a unit clause, then l must be assigned T. [8] To obtain this, KSAT_W is invoked recursively with arguments the wff returned by *assign*(l, φ) and the assignment obtained by adding l to μ. *assign*(l, φ) substitutes every occurrence of l in φ with T and evaluates the result.
- (split) If none of the above situations occurs, then *choose-literal*(φ) returns an unassigned literal l according to some heuristic criterion. Then KSAT_W is first invoked recursively with arguments *assign*(l, φ) and $\mu \wedge l$. If the result is negative, then KSAT_W is invoked with arguments *assign*$(\neg l, \varphi)$ and $\mu \wedge \neg l$.

$\text{KSAT}_A(\mu)$ invokes $\text{KSAT}_{RA}(\mu_r)$ (where "$_{RA}$" stands for *A*ssignment *R*estricted to one modality) for any index r such that $\Box_r$ occurs in μ. KSAT_{RA} returns a truth value asserting whether μ_r is K(m)-satisfiable or not.

The correctness and completeness of KSAT can be easily seen, for instance by noticing the close parallel with Fitting's tableau described in (Fitting 1983). It is important to notice that KSAT_W is a variant of the

[8] A notion of unit clause for non-CNF propositional wffs is given in (Armando & Giunchiglia 1993). More generally, (Armando & Giunchiglia 1993) and (Sebastiani 1994) show how decision procedures for CNF formulas can be modified to work for non-CNF formulas

non-CNF version of the Davis-Putnam-Longemann-Loveland SAT procedure (Davis & Putnam 1960, Davis, Longemann & Loveland 1962) (DPLL from now on), as described in (Armando & Giunchiglia 1993). Unlike DPLL, whenever an assignment μ has been found, KSAT$_W$, instead of returning *True*, invokes KSAT$_A(\mu)$. Essentially, DPLL is used to generate truth assignments, whose K(m)-satisfiability is recursively checked by KSAT$_A$. We have implemented the algorithm described in Figure 1 as a procedure, also called KSAT, implemented in Common Lisp on top of the non-CNF DPLL decision procedure described in (Armando & Giunchiglia 1993). DPLL is well known to be one of the fastest decision procedures for SAT (see, e.g., (Buro & Buning 1992, Uribe & Stickel 1994)). However the implementation we use, though relatively fast, is much slower than the state-of-the-art SAT decision procedures (see, e.g., (Buro & Buning 1992, Zhang & Stickel 1994)). The basic version of the algorithm described in Figure 1 is improved in the following way. First, all modal atoms are internally ordered. This avoids assigning different truth values to permutations of the same sub-wffs. Secondly, KSAT$_{RA}$ is implemented in such a way to "factorize" the common component $\bigwedge_i \alpha_{ri}$ in searching truth assignments for $\bigwedge_i \alpha_{ri} \wedge \neg\beta_{r1}$, $\bigwedge_i \alpha_{ri} \wedge \neg\beta_{r2}$, ... Finally, KSAT$_W$ is modified in such a way that KSAT$_A$ is (optionally) invoked on intermediate assignments before every split. This drastically prunes search whenever unconsistent intermediate assignments are detected. These topics are described in detail in (Giunchiglia & Sebastiani 1996). More recently we have also introduced a form of preprocessing — essentially, a recursive removal of duplicate and contradictory subwffs — of the input formulas.

3 THE TEST METHOD

The methodology we use generalizes the *fixed-clause-length* model commonly used in propositional SAT testing (see, e.g., (Mitchell et al. 1992, Buro & Buning 1992)).

Let a 3CNF$_{K(m)}$ wff be a conjunction of 3CNF$_{K(m)}$ clauses. Let a 3CNF$_{K(m)}$ clause be a disjunction of three 3CNF$_{K(m)}$ literals, i.e., 3CNF$_{K(m)}$ atoms or their negations. Let a 3CNF$_{K(m)}$ atom be either a propositional atom or a wff in the form $\Box_r C'$, C' being a 3CNF$_{K(m)}$ clause. Then 3CNF$_{K(m)}$ wffs are randomly generated according to the following parameters:

(i) the modal depth d;
(ii) the number of distinct boxes m;
(iii) the number of clauses L;
(iv) the number of propositional variables N;
(v) the probability p with which any randomly generated 3CNF$_{K(m)}$ atom is propositional. (p establishes thus the percentage of propositional atoms at every level of the wff tree.)

Notice that, if we set $d = 0$, we have the standard 3SAT test method (Mitchell et al. 1992).

For fixed N, d, m and p, for increasing values of L, a certain number (100, 500, 1000...) of random 3CNF$_{K(m)}$ wffs are generated, internally sorted, and then given in input to the procedure under test. Satisfiability percentages and mean/median CPU times are plotted against the L/N ratio.

Similarly to the propositional 3CNF case, the methodology proposed above presents three main features. First, the method is very general: 3CNF$_{K(m)}$ wffs represent all K(m) wffs, as there there is a K(m)-satisfiability-preserving way of converting any K(m) wff into 3CNF$_{K(m)}$. Second, the usage of 3CNF$_{K(m)}$ form minimizes the number of parameters to handle. Finally, the parameters L and N allow for a coarse "tuning" of both the satisfiability probability and the hardness of random 3CNF modal wffs, so that it is possible to generate very hard problems with near 0.5 satisfiability probability.

4 TABLEAU-BASED VS. SAT-BASED PROCEDURES

In the tests described in this section we have tested and compared TABLEAU, KRIS, KSAT$_0$ and KSAT on the same group of 4,000 random formulas, with $d = 2$, $m = 1$, $N = 3$, $p = 0.5$, $L \in \{N \ldots 40N\}$, 100 samples/point. These values have been chosen as in the analysis described in (Giunchiglia & Sebastiani 1996) they gave the highest execution times with both TABLEAU and KSAT$_0$. The range $N \ldots 40N$ for L has been chosen empirically to cover coarsely the "100% satisfiable - 100% unsatisfiable" transition. As a general test rule we have introduced a timeout of 1000s on each sample wff. If the decision procedure under test exceeds the timeout for a given wff, a failure value is returned and the CPU time value is conventionally set to 1000s. Furthermore, we have stopped running the whole test whenever more than 50% samples (e.g., 50 out of 100 samples) have taken more than 1000s each to execute. These two choices have caused a relevant reduction of the testing time. Figure 2 (left) presents the median CPU time plots for all four systems. (We compare median values rather

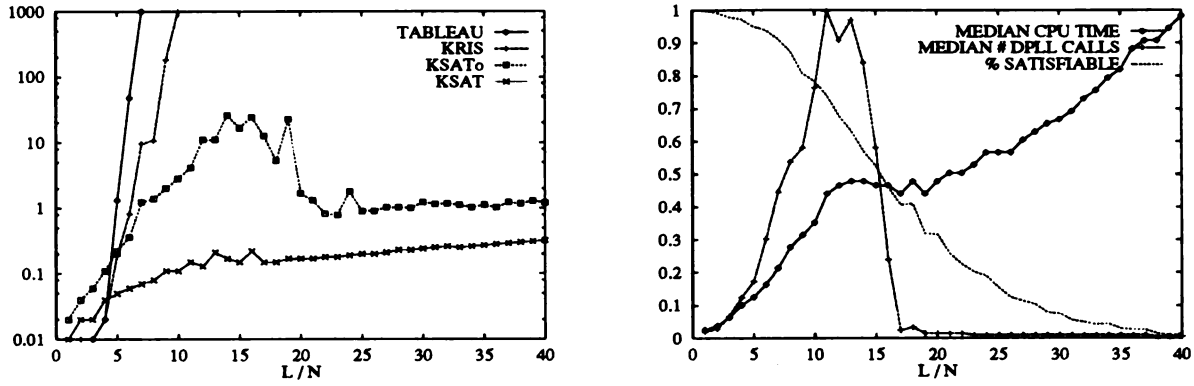


Figure 2: $d = 2$, $m = 1$, $N = 3$, $p = 0.5$, $L = N \ldots 40N$. Left: TABLEAU, KRIS, KSAT_0 and KSAT. Median CPU time, 100 samples/point. Right: KSAT. Normalized plots of Median CPU time, Median # of DPLL calls, satisfiability ratio, 1000 samples/point.

than mean values, as the former are much less sensitive to the noise introduced by outliers.) Notice the logarithmic scale on the vertical axis. In Figure 2 (right) we plot respectively the median CPU time, the median number of DPLL calls and the satisfiability percentage curves obtained by running KSAT on the same problem above, with 1000 sample wffs/point. In Figure 2 (right) the curves are all normalized to 1.[9]

Four observations can be made, given below in increasing order of importance.

First, improving the quality of the implementation, e.g., from TABLEAU to KRIS or from KSAT_0 to KSAT, may introduce good quantitative performance improvements. In fact, KRIS reaches the time bound at the 10th step, while TABLEAU reaches the time bound at the 7th step, about two orders of magnitude above the corresponding KRIS value. Similarly, KSAT_0 has a maximum at the 14th step, more than 2 orders of magnitude above the corresponding KSAT value. However, and this is the second observation, improving the quality of the implementation does not seem to affect the qualitative behaviour of the procedures. In fact, both the TABLEAU and KRIS curves present a supposedly exponential growth with the number of clauses, while both KSAT_0 and KSAT curves flatten when the number of clauses exceeds a certain value.

Third, independently from the quality of implementation, KSAT and KSAT_0 quantitatively outperform TABLEAU and KRIS. For instance, the performance gap between KSAT and KRIS at the 10th step is about 4 orders of magnitude. Moreover, the extrapolation of the KRIS curve suggests that its value — and the performance gap with KSAT — would reach several orders of magnitude for problems at the right end side of the plots. To support this consideration, we ran KRIS on 100 samples of the same problem, for $L = 40N$. No sample wff was solved within the timeout. When releasing the timeout mechanism, KRIS was not able to end successfully the computation of the first sample wff after a run of one month. Fourth, and most important, independently of the quality of implementation, KSAT and KSAT_0 qualitatively outperform TABLEAU and KRIS. In fact, while TABLEAU and KRIS present a supposedly exponential growth against the number of clauses, both KSAT_0 and KSAT curves present a polynomial growth. In particular, the KSAT CPU time curve (like that of KSAT_0) results from a combination of (i) a linear component and (ii) an easy-hard-easy component, centered in the satisfiability transition zone. Both components above are straightforward to notice in Figure 2 (right). The former is due to the preprocessing and to the linear-time function *assign*, which is invoked at every DPLL recursive call. The latter represents the number of recursive DPLL calls, i.e., the size of the tree effectively searched.

The quantitative and qualitative performance gaps pointed out by the third and the fourth observation above are very important and deserve some explanation. Let us consider for instance KSAT and KRIS. Both procedures work (i) by enumerating truth assignments which propositionally satisfy the input wff φ and (ii) by recursively checking the K(m)-satisfiability of the assignments found. Both algorithms perform the latter step in the same way. The key difference is

[9] The tests in Figures 2 (left) and 5 have been compiled and run under `Allegro CL 4.2` on a `SUN SPARC10 32M` workstation. The test in Figure 2 (right) has been compiled and run under `AKCL 1.623` on another `SUN SPARC10 32M` workstation. The tests in Figure 4 have been compiled and run under `Allegro CL 4.1` on two identical `SUN SPARC2 32M` workstations.

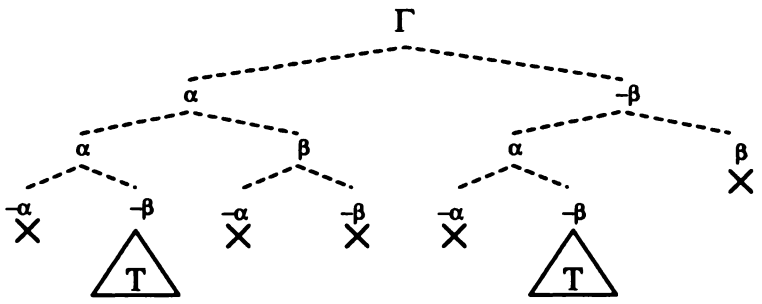


Figure 3: Tableau for the wff $\Gamma = (\alpha \vee \neg\beta) \wedge (\alpha \vee \beta) \wedge (\neg\alpha \vee \neg\beta)$.

in the first step, that is, in the way KSAT and KRIS handle propositional inference.

In KRIS truth assignments are (implicitly) generated as branches of an analytic propositional tableau, that is, by the recursive application of the rules:

$$\frac{\varphi_1 \wedge \varphi_2}{\varphi_1, \varphi_2}(\wedge\text{-rule}) \quad \frac{\varphi_1 \vee \varphi_2}{\varphi_1 \quad \varphi_2}(\vee\text{-rule}) \qquad (4)$$

and of the other rules for $\{\neg\vee, \neg\wedge, \supset, \neg\supset, \neg\neg\}$. Analytic propositional tableaux perform what we call *syntactic branching*, that is, a branching on the syntactic structure of φ. As widely discussed in (D'Agostino 1992, D'Agostino & Mondadori 1994), any application of the $\vee$-rule generates two subtrees which are *not mutually inconsistent*, [10] that is, two subtrees which may share propositional models. The set of truth assignments enumerated by propositional tableau procedures is intrinsically redundant, and may contain many duplicate and/or subsumed assignments. As a consequence, the number of truth assignments generated grows exponentially with the number of disjunctions occurring positively in φ (in our tests, the number of clauses L), although the actual number of non-redundant assignments propositionally satisfying φ is much smaller. This redundancy is a source of a high degree of inefficiency when using analytic tableaux for propositional satisfiability.

Things get much worse in the modal case. Unlike the propositional case — where tableaux look for *one* assignment satisfying the input formula — in K(m) propositional tableaux enumerate all the truth assignments, which must be recursively checked for K(m)-consistency. (The number of assignments may be huge: up to ten thousands in our tests.) This requires checking recursively (possibly many) subwffs of the form $\bigwedge_i \alpha_{ri} \wedge \beta_j$ of depth $d-1$, for which a propositional tableau will enumerate truth assignments, and so forth. Any redundant truth assignment enumerated at depth d introduces a redundant modal search tree of depth d. Even worse, this propositional redundancy propagates exponentially with the depth d, following the analysis of the subwffs of decreasing depth.

Example 4.1 Consider the simple wff

$$\Gamma = (\alpha \vee \neg\beta) \wedge (\alpha \vee \beta) \wedge (\neg\alpha \vee \neg\beta),$$

where α and β are modal atoms, and let d be the depth of Γ. The only possible assignment satisfying Γ is $\mu = \alpha \wedge \neg\beta$. Look at Figure 3. The $\vee$-rule is applied to the three clauses occurring in Γ in the order they are listed, and two distinct but identical open branches are generated, both representing the assignment μ. Suppose now that μ is not K(m)-consistent. Then the tableau expands the two open branches in the same way, until it generates two identical (and possibly big) closed modal sub-tableaux T of depth d, each proving the K(m)-inconsistency of μ. This phenomenon may repeat itself at the lower level in each sub-tableaux T, and so forth. For instance, if $\alpha = \Box((\alpha' \vee \neg\beta') \wedge (\alpha' \vee \beta'))$ and $\beta = \Box(\alpha' \wedge \beta')$, then at the lower level we have a wff Γ' of depth $d-1$ analogous to Γ. This propagates exponentially the redundancy with the depth d.

Notice that, if we considered the wff

$$\Gamma^K = \bigwedge_{i=1}^{K} (\alpha_i \vee \neg\beta_i) \wedge (\alpha_i \vee \beta_i) \wedge (\neg\alpha_i \vee \neg\beta_i),$$

the tableau would generate 2^K identical truth assignments $\mu^K = \bigwedge_i \alpha_i \wedge \neg\beta_i$, and things would get exponentially worse. □

In SAT-based procedures truth assignments are gen-

[10] As pointed out in (D'Agostino 1992, D'Agostino & Mondadori 1994), in Analytic tableaux rules are unable to represent *bivalence*: "every proposition is either true or false, *tertium non datur*". This is a consequence of the elimination of the cut rule in cut-free sequent calculi, from which analytic tableaux are derived.

erated one-shot by a SAT decision procedure. [11] SAT-based procedures perform a search based on what we call *semantic branching*, that is, a branching on the truth value of proper subwffs of φ. Every branching step generates two *mutually unconsistent* subtrees. Because of this, SAT procedures always generate non-redundant sets of assignments. This avoids any search duplication and, recursively on d, any exponential propagation of inefficiency.

Example 4.2 Consider the wff Γ in Example 4.1. A SAT-based procedure branches asserting $\alpha = T$ or $\alpha = F$. The first branch generates $\alpha \wedge \neg\beta$, while the second gives $\neg\alpha \wedge \neg\beta \wedge \beta$, which immediately closes. Therefore, only one instance of the assignment $\mu = \alpha \wedge \neg\beta$ is generated. The same applies recursively to μ^K. □

A propositional wff φ can be seen in terms of a set of constraints for the truth assignments which possibly satisfy it (see, e.g., (Williams & Hogg 1994)). For instance, a clause $A_1 \vee A_2$ constrains every assignment not to set both A_1 and A_2 to F. Unlike tableaux, in SAT procedures branches are cut as soon as they violate some constraint of the wff. The more constrained the wff is, the more likely a truth assignment violates some constraint. (For instance, the bigger is L in a CNF wff, the more likely an assignment generates an empty clause.) Therefore, as φ becomes highly constrained (e.g., when L is big enough) the search tree is very heavily pruned. As a consequence, for L bigger than a certain value, the size of the search tree *decreases* with L, as it can be easily noticed in Figure 2 (right).

5 AN EXAUSTIVE EMPIRICAL ANALYSIS

In the tests described in this section we have tested and compared KSAT and KRIS, that is, the fastest SAT-based and the fastest tableau-based decision procedure at our disposal. We have performed three experiments on 48,000 randomly generated wffs, run according to our test methodology, whose results are all described in Figure 4. All curves represent 100 samples/point. As above, the range $N \ldots 40N$ for L has been chosen empirically to cover coarsely the "100% satisfiable – 100% unsatisfiable" transition. In each experiment we investigate the effects of varying one parameter while fixing the others. In Experiment 1 (left column) we fix $d = 2$, $m = 1$, $p = 0.5$ and plot different curves for increasing numbers of variables $N = 3, 4, 5$. [12] In Experiment 2 (center column) we fix $d = 2$, $N = 4$, $p = 0.5$ and plot different curves for increasing number of distinct modalities $m = 1, 2, 5, 10, 20$. In Experiment 3 (right column) we fix $m = 1$, $N = 3$, $p = 0.5$ and plot different curves for increasing modal depths $d = 2, 3, 4, 5$. For each experiment, we present three distinct sets of curves, each corresponding to a distinct row. In the first (top row) we plot the median CPU time obtained by running both KSAT and KRIS. This gives an overall picture of the qualitative behaviour of KSAT and KRIS and allows for a direct comparison between them. In the second (middle row) we plot the KSAT median number of recursive DPLL calls, that is, the size of the space effectively searched. This allows us to drop the linear component due to the preprocessing and the function calls to *assign*. In the third (bottom row) we plot the percentage of satisfiable wffs evaluated by KSAT. This gives a coarse indication of the average level of constraintness of the test wffs. [13]

Despite the big noise, due to the small samples/point rate (100), the results indicated in Figure 4 provide interesting indications. We report below (Subsection 5.1) a first pass, experiment by experiment, analysis of the results. This gives us an idea of how efficiency and satisfiability are affected by each single parameter. In Subsection 5.2 we report a global and, in some respects, more interesting analysis of the results we have.

5.1 A TESTWISE ANALYSIS

The results of the first experiment (left column) show that increasing N (and L accordingly) causes a relevant increase in complexity — up to one order of magnitude per variable in the "hard" zone for KSAT, up to two orders of magnitude per variable, as far as we can see, for KRIS. This should not be a surprise, as in K/K(m), adding few variables may cause an ex-

[11] In KSAT we used non-CNF DPLL, but we could use any other SAT procedures not affected by the problem highlighted in (D'Agostino 1992, D'Agostino & Mondadori 1994), e.g., OBDDs (Bryant 1992), or an implementation of KE (D'Agostino & Mondadori 1994).

[12] If we compare the KSAT and KRIS plots in Figure 2 (left) with the $L = 3$ KSAT and KRIS plots in Figure 4 (top left), we notice that the plots are different, although they are computed on sample wffs with the same parameter values. This is due to the fact that (i) the former ones are run on a much faster machine; (ii) the starting seeds are different, causing thus the generation of different sample sets.

[13] This percentage is evaluated on the number of samples which *effectively* ended computation within the timeout. Therefore this datum should be considered only as a coarse indication. To obtain an accurate evaluation, we should drop the timeout mechanism and evaluate the satisfiability percentage on at least 1000 samples/point, like in Figure 2 (right).

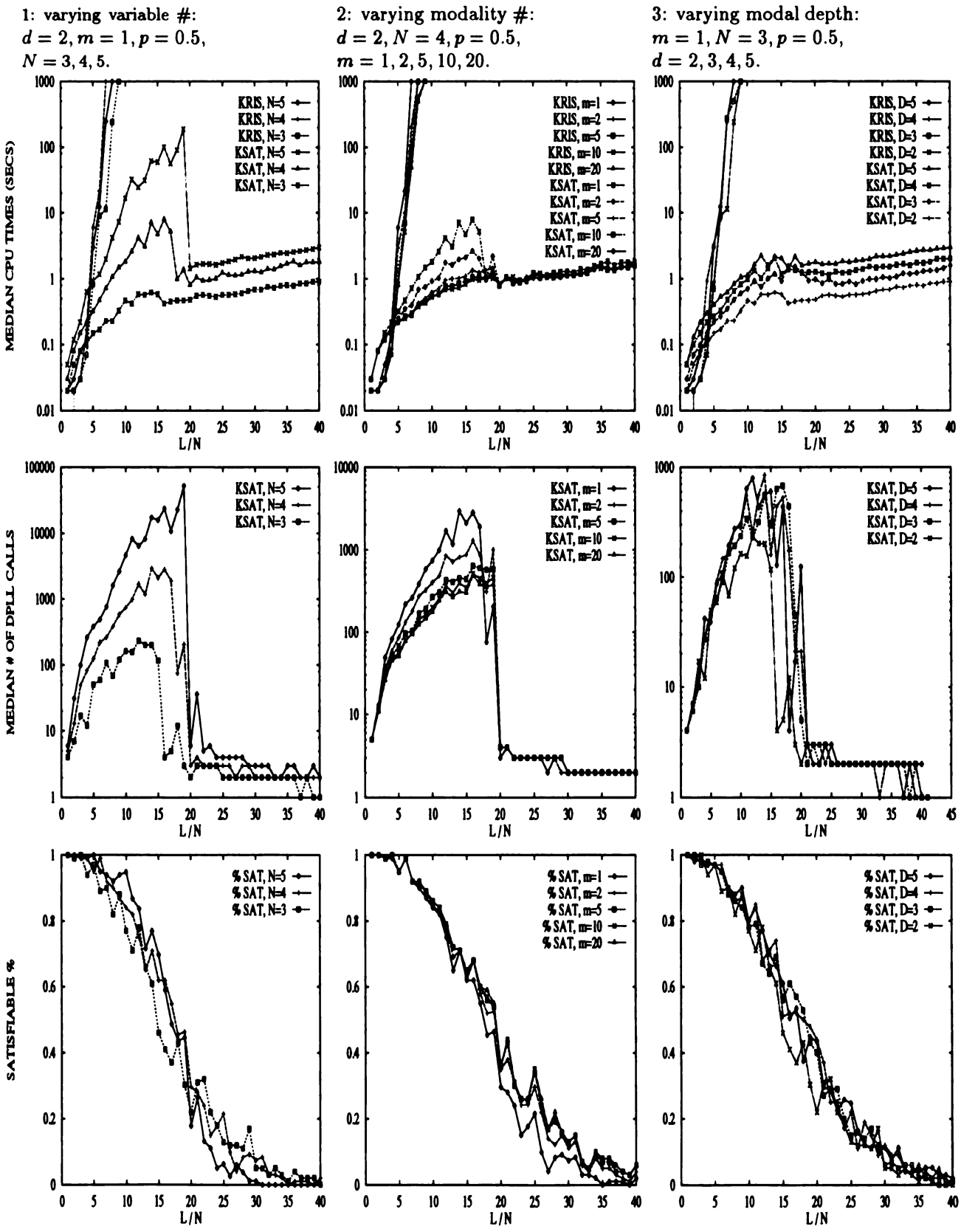


Figure 4: The results of the three experiments.

ponential increase of the search space. Each variable may in fact assume distinct truth values inside distinct states/possible worlds, that is, each variable must be considered with an "implicit multiplicity" equal to the number of states of a potential Kripke model.

The results of the second experiment (center column) present two interesting aspects. First, the complexity of the search monotonically decreases with the increase of the number m of modalities, for both KSAT and KRIS (top and middle box). At a first sight it may sound like a surprise, but it should not be so. In fact, each truth assignment μ is partitioned into m independent sub-assignments μ_r's, each restricted to a single $\Box_r$ (see Equations (1) and (2)). This means "dividing and conquering" the search tree into m non-interfering search trees. Therefore, the bigger is m, the more partitioned is the search space, and the easier is the problem to solve. Second, a careful look reveals that the satisfiability percentage increases with m. Again, there is no mutual dependency between the satisfiability of the distinct μ_r's. Therefore the bigger is m, the less constrained is μ, and the more likely satisfiable is φ.

The results of the third experiment (right column) provide evidence of the fact that the complexity increases with the modal depth d, for both KSAT and KRIS. This is rather intuitive: the higher is d, the deeper are the Kripke models to be searched, and the higher is the complexity of the search.

5.2 A GLOBAL ANALYSIS

The KSAT curves (top and middle rows) highlight the existence a linear and an easy-hard-easy component. In fact, if we increase N from 3 to 5 and L accordingly (left column), the size of the search space has a relevant increase. Therefore, while for $N = 3$ the linear component prevails, for $N = 5$ the easy-hard-easy component dominates. Moreover, when varying the number of modalities (center column), the wff sizes are kept the same for all curves. Therefore, when the effect of the easy-hard-easy component vanishes ($L/N > 20$), the curves collapse together, as the time for preprocessing and *assign* does not depend on the number of modalities m. Notice that the locations of the easy-hard-easy zones do not seem to vary significantly, neither with the number of variables N (left column), nor with the number of modalities m (center column), nor with the depth d (right column).

Let us now consider the satisfiability plots in Figure 4 (bottom row). Despite the noise and the approximations due to timeouts, it is easy to notice that the 50% satisfiability point is centered around $L = 15N \sim 20N$

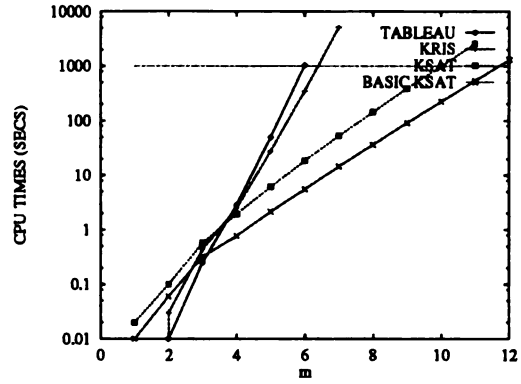


Figure 5: CPU times for the class of φ_d^K formulas.

in all the experiments. Moreover, in the first experiment a careful look reveals that the satisfiability transition becomes steeper when increasing N (e.g., compare the $N = 3$ and $N = 5$ plots). Finally, in all experiments, the curves representing the median number of DPLL calls (middle row) generally locate the peaks around the satisfiability transition, although they seem to anticipate a little the 50% crossover point. From these facts we may conjecture (to be verified!) the existence for K(m)/$\mathcal{ALC}$ of a phase transition phenomenon, similar to that already known for SAT and other NP-hard problems (see, e.g, (Cheeseman, Kanefski & Taylor 1991, Mitchell et al. 1992, Kirkpatrick & Selman 1994)).

The final observation comes from the three sets of median CPU times curves (top row): KSAT outperforms KRIS in all the testbeds, independently on the number of variables N, the number of modalities m or the depth d considered. This confirms the analysis done in Section 4. Again, this is not only a quantitative performance gap (up to 3-4 orders of magnitude) but also a qualitative one, as all KRIS curves grow (supposedly) exponentially with L, while all KSAT curves grow polinomially. To provide further evidence, we have performed another, quite different, test, based on the class of wffs $\{\varphi_d^K\}_{d=1,2,\ldots}$ presented in (Halpern & Moses 1992). This is a class of K(1)-satisfiable wffs, with depth d and $2d+1$ propositional variables. These wffs are paradigmatic for modal K, 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 & Moses 1992) we can reasonably assume a minimum exponential growth factor of 2^d for any ordinary algorithm based on Kripke semantics. We run TABLEAU, KRIS, KSAT and a "basic" version of KSAT (i.e., with no factorization of $\bigwedge_i \alpha_{ri}$'s and no checking of intermediate assignments), called below BASIC KSAT, on these formulas, for increasing values of d.

The results are plotted in Figure 5. The TABLEAU, KRIS, KSAT and BASIC KSAT curves grow exponentially, approximatively as $(16.0)^d$, $(12.7)^d$, $(2.6)^d$ and $(2.4)^d$ respectively, exceeding 1000s for $d = 6$, $d = 7$, $d = 11$ and $d = 12$ respectively. The slight difference between KSAT and BASIC KSAT is due to the overhead introduced by the $\bigwedge_i \alpha_{ri}$ factorization, which is useless with these formulas. It is worth observing that the result of tracing the global number of truth assignments μ, recursively found by both KSAT and BASIC KSAT, gave exactly $2^{d+1} - 1$ for every d, that is the minimum number of Kripke states. KSAT and BASIC KSAT found no redundant truth assignments.

6 CONCLUSIONS

This paper presents what we think are three very important results:

1. it provides a new implemented algorithm, KSAT, for deciding satisfiability in $\mathcal{ALC}$ (K(m)) which outperforms of orders of magnitude the previous state-of-the-art decision procedures;
2. it shows that the results provided are not by chance, and that all SAT-based modal decision procedures (that is, all the modal decision procedures based on SAT decision procedures) are intrinsically bound to be more efficient than tableau-based decision procedures; and
3. it provides evidence, though very partial, of an easy-hard-easy pattern independent of all the parameters of evaluation considered. If the current partial evidence is confirmed, this is the first time that this phenomenon, well known for SAT and other NP-hard problems, is found in modal logics.

Acknowledgements

Franz Baader, Marco Cadoli, Enrico Franconi, Enrico Giunchiglia, Fabio Massacci and Bernhard Nebel have given very useful feedback. Fabio Massacci has suggested testing the Halpern & Moses formulas. Marco Roveri has given technical assistance in the testing phase. All the members of the Mechanized Reasoning Group in Genoa have put up with many weeks of CPU-time background processes.

References

Armando, A. & Giunchiglia, E. (1993), 'Embedding Complex Decision Procedures inside an Interactive Theorem Prover', *Annals of Mathematics and Artificial Intelligence* **8**(3–4), 475–502.

Baader, F., Franconi, E., Hollunder, B., Nebel, B. & Profitlich, H. (1994), '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.

Bryant, R. E. (1992), 'Symbolic Boolean manipolation with ordered binary-decision diagrams', *ACM Computing Surveys* **24**(3), 293–318.

Buro, M. & Buning, H. (1992), Report on a SAT competition, Technical Report 110, University of Paderborn, Germany.

Cheeseman, P., Kanefski, B. & Taylor, W. M. (1991), Where the really hard problem are, *in* 'Proc. of the 12th International Joint Conference on Artificial Intelligence', pp. 163–169.

D'Agostino, M. (1992), 'Are Tableaux an Improvement on Truth-Tables?', *Journal of Logic, Language and Information* **1**, 235–252.

D'Agostino, M. & Mondadori, M. (1994), 'The Taming of the Cut.', *Journal of Logic and Computation* **4**(3), 285–319.

Davis, M., Longemann, G. & Loveland, D. (1962), 'A machine program for theorem proving', *Journal of the ACM* **5**(7).

Davis, M. & Putnam, H. (1960), 'A computing procedure for quantification theory', *Journal of the ACM* **7**, 201–215.

Fitting, M. (1983), *Proof Methods for Modal and Intuitionistic Logics*, D. Reidel Publishg.

Giunchiglia, F. & Sebastiani, R. (1996), 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, Springer–Verlag.

Giunchiglia, F. & 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. & 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.

Halpern, J. & Moses, Y. (1992), 'A guide to the completeness and complexity for modal logics of knowledge and belief', *Artificial Intelligence* **54**(3), 319–379.

Hollunder, B., Nutt, W. & Schmidt-Schauß, M. (1990), Subsumption Algorithms for Concept Description Languages, *in* 'Proc. 8th European Conference on Artificial Intelligence', pp. 348–353.

Kirkpatrick, S. & Selman, B. (1994), 'Critical behaviour in the satisfiability of random boolean expressions', *Science* **264**, 1297–1301.

Massacci, F. (1994), Strongly analytic tableaux for normal modal logics, *in* 'Proc. of the 12th Conference on Automated Deduction'.

Mitchell, D., Selman, B. & Levesque, H. (1992), Hard and Easy Distributions of SAT Problems, *in* 'Proc. of the 10th National Conference on Artificial Intelligence', pp. 459–465.

Schild, K. D. (1991), A correspondence theory for terminological logics: preliminary report, *in* 'Proc. of the 12th International Joint Conference on Artificial Intelligence', Sydney, Australia, pp. 466–471.

Schmidt-Schauß, M. & Smolka, G. (1991), 'Attributive Concept Descriptions with Complements', *Artificial Intelligence* **48**, 1–26.

Sebastiani, R. (1994), 'Applying GSAT to Non-Clausal Formulas', *Journal of Artificial Intelligence Research* **1**, 309–314. Also DIST-Technical Report 94-0018, DIST, University of Genova, Italy.

Uribe, T. E. & Stickel, M. E. (1994), Ordered Binary Decision Diagrams and the Davis-Putnam Procedure, *in* 'Proc. of the 1st International Conference on Constraints in Computational Logics'.

Williams, C. P. & Hogg, T. (1994), 'Exploiting the deep structure of constraint problems', *Artificial Intelligence* **70**, 73–117.

Zhang, H. & Stickel, M. (1994), Implementing the Davis-Putnam algorithm by tries, Technical report, University of Iowa.

Description Logics: Expressivity and Complexity

TBox and ABox Reasoning in Expressive Description Logics

Giuseppe De Giacomo
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
degiacomo@dis.uniroma1.it

Maurizio Lenzerini
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
lenzerini@dis.uniroma1.it

Abstract

A Description Logic (DL) system is characterized by four fundamental aspects: the set of constructs used in concept and role expressions, the kind of assertions allowed in the TBox (assertions on concepts) and the ABox (assertions on individuals), and the inference mechanisms for reasoning on both the TBox and the ABox. Most of the research done in the last decade made several simplifying assumptions on the above aspects. However, the recent interest in DLs exhibited in many application areas (databases, software engineering, intelligent access to the network, planning, etc.) calls for investigating DL systems with full capabilities. The work presented in this paper represents a step in this direction. We present a sound, complete, and terminating (in worst-case EXPTIME) inference procedure that solves the problem of reasoning in a DL system with the following characteristics: it comes equipped with a very expressive language, it allows the most general form of TBox assertions, and it takes into account instance assertions on both concepts and roles in the ABox.

1 INTRODUCTION

The research on Knowledge Representation has always paid attention to languages for the representation of classes and relationships. Description Logics (DLs) have been studied in the last decade as a formalization of these languages (see (Woods & Schmolze, 1992; Donini, Lenzerini, Nardi, & Schaerf, 1996; Borgida & Patel-Schneider, 1994; Baader, Hollunder, Nebel, Profitlich, & Franconi, 1992)). They allow one to represent a domain of interest in terms of *concepts* and *roles*, where concepts model classes of individuals, and roles model relationships between classes. Starting with atomic concepts and atomic roles, which are simply described by a name, complex concepts and roles can be denoted by expressions built using suitable constructs. Concepts and roles are given Tarskian semantics in terms of sets and binary relations, respectively.

A knowledge base expressed in a DL is constituted by two components, traditionally called TBox and ABox. The TBox stores a set of universally quantified assertions (inclusion assertions) stating general properties of concepts and roles. For example, an assertion of this kind is the one stating that a concept represents a specialization of another concept. The ABox comprises assertions on individual objects (instance assertions). A typical assertion in the ABox is the one stating that an individual is an instance of a certain concept.

Several reasoning tasks can be carried out on a knowledge base of the above kind. The simplest form of reasoning involves computing the subsumption relation between two concept expressions, i.e. verifying whether one expression always denotes a subset of the objects denoted by another expression. A more complex reasoning task consists in checking whether a certain assertion (either an inclusion or an instance assertion) is logically implied by a knowledge base.

A DL system is then characterized by four aspects:

1. The set of constructs constituting the language used for building the concepts and the roles mentioned in the TBox and in the ABox.
2. The kind of assertions that may appear in the TBox.
3. The kind of assertions that may appear in the ABox.
4. The inference mechanisms provided for reasoning on the knowledge bases expressible in the system.

It follows that the expressive power and the deduction capabilities of a DL system depends on the various choices and assumptions that the system adopts with regard to the above aspects. As to the fourth aspect, we concentrate in this paper on inference mechanisms that are sound and complete with respect to the standard semantics, although other choices are possible (see (Patel-Schneider, 1989)).

Most of the basic research work on the computational complexity of DLs has been carried out in a simplified context where both the TBox and the ABox are empty (see (Donini, Lenzerini, Nardi, & Nutt, 1991a, 1991b; Nebel, 1988)). This is not surprising, since these works aimed at studying the language constructs in isolation, with the goal of singling out their impact on the complexity of subsumption between concept expressions.

Other papers dealt with logical implication of ABox assertions under the simplifying assumption of an empty TBox, again with the goal of studying how the various language constructs influence the reasoning on individuals (see (Donini, Lenzerini, Nardi, & Schaerf, 1994; Schaerf, 1994)).

More recently, there has been a strong interest in the problem of reasoning with TBox assertions in isolation (see (De Giacomo & Lenzerini, 1994a; Calvanese, De Giacomo, & Lenzerini, 1995; Nebel, 1991; Baader, 1991; Schild, 1994)). One important outcome of this line of research is that, limiting the expressive power of the language with the goal of gaining tractability is useless in this setting, because the power of TBox assertions alone (when no limitations on cycles in the TBox are imposed) generally leads to high complexity in the inference mechanisms. For this reason, these investigations often refer to very powerful languages for expressing concepts and roles.

The complete setting has been the subject of some investigations only recently. For example, in (Buchheit, Donini, & Schaerf, 1993) a DL system with both the TBox and the ABox is studied with a relatively powerful language (not including inverse roles) . However, results about reasoning on knowledge bases with both the TBox and the ABox are still rare.

We observe that such results would be very important in the light of the renewed interest in DLs that we find in disparate application areas. Indeed, DL systems are now advocated as suitable knowledge representation systems in many contexts, such as information systems (Catarci & Lenzerini, 1993), databases (Borgida, 1995; Bergamaschi & Sartori, 1992; Sheth, Gala, & Navathe, 1993), software engineering (Devambu, Brachman, Selfridge, & Ballard, 1991), intelligent access to the network (Levy, Rajaraman, & Ordille, 1996; Blanco, Illarramendi, & Goni, 1994), action representation (Artale & Franconi, 1994), and planning (Weida & Litman, 1992). Many of the above papers point out that the whole capabilities of a DL system (expressive language, TBox and ABox assertions) are often required in the corresponding application fields (see also (Doyle & Patil, 1991)).

The work presented in this paper represents a fundamental step in this direction. We present a sound, complete, and terminating inference procedure that solves the problem of reasoning in a DL system with the following characteristics:

1. It comes equipped with a very expressive language, comprising all classical concept forming constructs, plus several role forming constructs (including inverse roles), and the most general form of number restrictions.
2. It allows the most general form of TBox assertions, without any limitations on the presence of cycles.
3. It allows expressing instance assertions on both concepts and roles in the ABox.

The most important contributions of our work can be summarized as follows:

- We present the first decidability result for a DL system combining inverse roles, number restrictions, and TBox and ABox assertions simultaneously.
- We present the first technique for reasoning on ABox assertions in a DL system that does *not* enjoy the finite model property (a knowledge base in our system may have only models with infinite domains).
- Our technique is optimal with respect to the complexity class of the inference problem (EXPTIME), and has the same computational complexity (in the worst case) as the procedure for reasoning in a TBox expressed in the basic language $\mathcal{ALC}$ (Schmidt-Schauß & Smolka, 1991).

The paper is organized as follows. In Section 2, we present the DL language we are interested in, called $\mathcal{CIQ}$. In Section 3 we illustrate the various features of $\mathcal{CIQ}$ by means of some examples. In Section 4 we briefly discuss the correspondence between DLs and propositional dynamic logics (PDLs) which is at the base of our results on the reasoning procedures for

$\mathcal{CIQ}$. In Section 5 we introduce some technical notions that will be needed to get our results. In Section 6, we describe our technique for computing logical implication over knowledge bases built using $\mathcal{CIQ}$. Finally, in Section 7 we draw some conclusions.

2 $\mathcal{CIQ}$ SYNTAX AND SEMANTICS

In the following, we focus on the description logic $\mathcal{CIQ}$ which has been studied in (De Giacomo & Lenzerini, 1995; De Giacomo, 1995). The available constructs for concept and role expressions in $\mathcal{CIQ}$ are specified in Figure 1.

Note that $\mathcal{CIQ}$ is a very expressive language, comprising all usual concept constructs, including the most general form of number restrictions, the so called *qualified number restrictions*, and a rich set of role constructs, namely: *union* of roles $R_1 \sqcup R_2$, *chaining* of roles $R_1 \circ R_2$, *reflexive-transitive* closure of roles R^*, *inverse* roles R^-, and the identity role $id(C)$ projected on C also called *test* in the following.

The semantics of $\mathcal{CIQ}$ interprets concepts as subsets of a domain, and roles as binary relations over such a domain. Formally, an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a *domain of interpretation* $\Delta^{\mathcal{I}}$, and an *interpretation function* $\cdot^{\mathcal{I}}$ mapping every atomic concept A to a subset of $\Delta^{\mathcal{I}}$, and every atomic role P to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function is systematically extended to complex concepts and roles according to the semantics of the constructs given in Figure 1. In the figure, $\sharp S$ denotes the cardinality of the set S, and $(R^{\mathcal{I}})^i$ stands for i repetitions of $R^{\mathcal{I}}$ – i.e., $(R^{\mathcal{I}})^0 = (id(\top))^{\mathcal{I}}$, and $(R^{\mathcal{I}})^i = R^{\mathcal{I}} \circ (R^{\mathcal{I}})^{i-1}$.

A $\mathcal{CIQ}$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is constituted by two components: a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$.

The TBox is a finite set of *inclusion assertions* of the form:

$$C_1 \sqsubseteq C_2$$

where C_1 and C_2 are concepts. In the following we use $C \equiv D$ as an abbreviation of $C \sqsubseteq D$ and $D \sqsubseteq C$.

The ABox is a finite set of *instance assertions* of the form:

$$C(\alpha)$$

where C is a concept, and α is an individual name, or of the form:

$$P(\alpha_i, \alpha_j)$$

where P is a primitive role and α_i, α_j two individuals names. An interpretation $\mathcal{I}$ maps individual names to individuals in $\Delta^{\mathcal{I}}$, in such a way that different individual names denote different individuals. Therefore we do not make any distinction between individuals and their names in the following.

An interpretation $\mathcal{I}$ is a model of an inclusion assertion $C_1 \sqsubseteq C_2$ if $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a model of an instance assertion $C(\alpha)$ if $\alpha \in C^{\mathcal{I}}$, and is a model of $P(\alpha_i, \alpha_j)$ if $(\alpha_i, \alpha_j) \in P^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a model of knowledge base $\mathcal{K}$ if $\mathcal{I}$ is a model of each inclusion and instance assertion in $\mathcal{K}$. $\mathcal{K}$ is satisfiable if it has a model. $\mathcal{K}$ logically implies an (inclusion or instance) assertion σ, written $\mathcal{K} \models \sigma$, if σ is satisfied by every model of $\mathcal{K}$. A concept C is satisfiable in $\mathcal{K}$ if there is a model $\mathcal{I}$ of $\mathcal{K}$ such that $C^{\mathcal{I}} \neq \emptyset$. Observe that satisfiability of a concept C in a knowledge base $\mathcal{K}$ can be reformulated in terms of logical implication as $\mathcal{K} \not\models C \sqsubseteq \bot$, and in terms of satisfiability of a knowledge base as the satisfiability of $\mathcal{K} \cup \{C(\alpha_{new})\}$, where α_{new} in an individual not mentioned in $\mathcal{K}$.

3 EXAMPLES

Figure 2 shows a $\mathcal{CIQ}$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, concerning directories and files. The TBox $\mathcal{T}$ is made by four assertions.

The first inclusion assertion states that every `dir_child` of an instance d of `Directory` is either a directory or a file, and has exactly one `dir_child`$^-$ predecessor, which is d itself. In other words, the fragment of `dir_child` starting from an instance of `Directory` has a structure similar to a tree, except that cycles are not prevented.

The second inclusion assertion states that instances of `File` have no children, and that are distinct from instances of `Directory`.

The third assertion states that the instances of `FileSysRoot` are directories which have no `dir_child`-predecessor.

The forth assertion states that every instance of `FileSysElement` reaches an instance of `FileSysRoot` in a finite number of steps through a chain of `dir_child`$^-$. It can be seen that such constraint, together with the first assertion, prevents cycles to appear in `dir_child`-chains involving instances of `FileSysElement`.

The ABox $\mathcal{A}$ can be thought of as divided into parts.

The first part is made of instance assertions concerning the individuals `a`, `b` and `c`, and their `dir_child` relationships. It expresses that `a` has two children, namely `b` and `c`, and that `a` is in turn a child of `c` (i.e. there is a cycle involving `a` and `b`).

The second part concerns the individuals `MyDir`,

Construct Name	Syntax	Semantics
atomic concept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
top	$\top$	$\Delta^{\mathcal{I}}$
bottom	$\bot$	$\emptyset$
conjunction	$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
disjunction	$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
negation	$\neg C$	$\Delta^{\mathcal{I}} - C^{\mathcal{I}}$
existential quantification	$\exists R.C$	$\{s \mid \exists s'.(s,s') \in R^{\mathcal{I}} \text{ and } s' \in C^{\mathcal{I}}\}$
universal quantification	$\forall R.C$	$\{s \mid \forall s'.(s,s') \in R^{\mathcal{I}} \text{ implies } s' \in C^{\mathcal{I}}\}$
qualified number restrictions	$(\geq n\, Q.C)$	$\{s \mid \sharp\{s'.(s,s') \in Q^{\mathcal{I}} \text{ and } s' \in C^{\mathcal{I}}\} \geq n\}$
	$(\leq n\, Q.C)$	$\{s \mid \sharp\{s'.(s,s') \in Q^{\mathcal{I}} \text{ and } s' \in E^{\mathcal{I}}\} \leq n\}$
atomic role	P	$P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
union	$R_1 \sqcup R_2$	$R_1^{\mathcal{I}} \cup R_2^{\mathcal{I}}$
chaining	$R_1 \circ R_2$	$R_1^{\mathcal{I}} \circ R_2^{\mathcal{I}}$
reflexive-transitive closure	R^*	$\bigcup_{i>0}(R^{\mathcal{I}})^i$
test	$id(C)$	$\{(s,s) \mid s \in C^{\mathcal{I}}\}$
inverse	R^-	$\{(s,s') \mid (s',s) \in R^{\mathcal{I}}\}$
basic role	$Q = P \mid P^-$	

Figure 1: Syntax and semantics of $\mathcal{CIQ}$ concept and role constructs.

TBox:	ABox:
$\texttt{Directory} \sqsubseteq \forall \texttt{dir\_child}.((\texttt{Directory} \sqcup \texttt{File}) \sqcap (\leq 1\, \texttt{dir\_child}^-.\top))$ $\texttt{File} \sqsubseteq (\forall \texttt{dir\_child}.\bot) \sqcap \neg \texttt{Directory}$ $\texttt{FileSysRoot} \sqsubseteq \texttt{Directory} \sqcap \forall \texttt{dir\_child}^-.\bot$ $\texttt{FileSysElement} \equiv \exists (\texttt{dir\_child}^-)^*.\texttt{FileSysRoot}$	$\texttt{dir\_child(a, b)}$ $\texttt{dir\_child(a, c)}$ $\texttt{dir\_child(c, a)}$ $\texttt{dir\_child(MyDir, Research)}$ $\texttt{dir\_child(MyDir, Teaching)}$ $\texttt{dir\_child(Research, CIQ.tex)}$ $\texttt{FileSysElement(CIQ.tex)}$ $\texttt{File(CIQ.tex)}$

Figure 2: Example: directories and files.

Teaching, Research and CIQ.tex. It expresses that MyDir has Teaching and Research as children, and that CIQ.tex is a child of Research. Moreover CIQ.tex is both a FileSysElement and a File.

From $\mathcal{K}$ we can make the following inference:

$$\mathcal{K} \models \texttt{FileSysElement} \sqsubseteq \texttt{Directory} \sqcup \texttt{File}.$$

Let us prove the above logical implication. By the fourth assertion in the TBox, every instance s of FileSysElement reaches an instance s' of FileSysRoot in a finite (but indeterminate) number n of $\texttt{dir\_child}^-$ steps. We proceed by *induction* on n. If $n = 0$, then $s = s'$. Hence s is an instance of FileSysRoot and so is an instance of Directory. If $n = k+1 > 0$, then let s'' be the immediate dir_child-predecessor along the chain. By induction hypothesis, s'' either a directory or a file. Since it has a dir_child-successor, namely s, it must be a directory and hence all is children, including s are either a directory or a file.

With minimal modification, this proof applies to the following logical implication as well:

$$\mathcal{K} \models \texttt{FileSysElement} \sqsubseteq \forall (\texttt{dir\_child}^-)^*.\texttt{Directory}.$$

The knowledge base K logically implies also that a, b, and c are not FileSysElement, i.e. for $\alpha = \texttt{a}, \texttt{b}, \texttt{c}$:

$$\mathcal{K} \models \neg \texttt{FileSysElement}(\alpha).$$

To prove the above logical implication we may reason as follows. First observe that none of a, b, and c

can be an instance of FileSysRoot, since all of them have an immediate dir_child-predecessor. Now let, for example, α = b. Suppose that b is an instance of FileSysElement. By the fourth assertion in the TBox, b must be connected by a finite chain of $\mathtt{dir\_child}^-$ to an instance of FileSysRoot. Also, as we saw before, the fact that b is an instance of FileSysElement implies that all dir_child-predecessors of b are instances of Directory. It follows from the first assertion in the TBox that each of the individuals a, b, and c has at most one dir_child immediate predecessor, and therefore, there are no dir_child-predecessors of b other than a and c. Since neither a nor c can be an instance of FileSysRoot, we have a contradiction. Hence we can conclude that neither a, b, or c are instances of FileSysElement.

With similar reasoning we can prove that, for β = MyDir, Teaching, Research:

$$\mathcal{K} \models \mathtt{FileSysElement}(\beta)$$

and also that MyDir and Research are instances of Directory, while Teaching is either an instance of Directory or an instance of File. In addition, we can prove that it is consistent that MyDir is an instance of FileSysRoot, though it is not logically implied.

Observe that in the proofs above, the use of induction is essential. Thus, the automatic reasoning procedure for $\mathcal{CIQ}$ must include either implicitly or explicitly such form of induction. The need for induction comes, as shown in the examples, from the presence of the reflexive-transitive closure of roles, which allows the specification of properties of objects that are distant a finite but indeterminate number of steps away (through a chain of roles). This ability testifies the non-first-order nature of our logic. $\mathcal{CIQ}$ is indeed a subset of *first order logic + fixpoints* (see (De Giacomo & Lenzerini, 1994b) and not of the pure *first order logic*, as most description logics are.

Finally, note that the knowledge base described in Figure 2 enjoys the finite model property. However, it easy to build a $\mathcal{CIQ}$ knowledge base with only infinite models. For example consider the following knowledge base:

TBox:	$\top \sqsubseteq (\leq 1\,\mathtt{succ}.\top) \sqcap (\leq 1\,\mathtt{succ}^-.\top)$
	$\mathtt{InfSeq} \sqsubseteq \exists \mathtt{succ}.\mathtt{InfSeq}$
ABox:	$(\mathtt{InfSeq} \sqcap \forall \mathtt{succ}^-.\bot)(\mathtt{Init})$

The first assertion constrains the role succ and its inverse to be functional.

The second assertion constrains the instances of InfSeq to have *its* immediate successor in InfSeq as well, i.e. each instance of InfSeq either is a (not necessarily immediate) successor of itself, or has an infinite chain of successors.

The instance assertion states that the individual Init is an instance of InfSeq but does not have any predecessor.

Now, since Init has no predecessor, it cannot be a successor of itself, so being an InfSeq it must have an infinite chain of successor. Hence all models of the knowledge base are infinite.

Observe that the existence of knowledge bases that admit only infinite models makes the reasoning methods based on the direct search and construction of a model (as the tableaux-based method in (Donini et al., 1994)) infeasible for $\mathcal{CIQ}$. Any reasoning procedure for $\mathcal{CIQ}$ based on the construction of a model, may at most construct a finite structure that represents a model (a pseudo-model), in the sense that it contains enough information so that, in principle, it can be expanded (maybe not univocally) to a model.

4 CORRESPONDENCE WITH PDLs

In the next sections, we will describe the procedure for reasoning in $\mathcal{CIQ}$-knowledge bases. Such procedure is based on the inference technique that the authors developed for the description logic $\mathcal{CIQ}$, mainly based on the correspondence between DLs and Propositional Dynamic Logics (PDLs) (Schild, 1991; De Giacomo & Lenzerini, 1994a; De Giacomo, 1995). PDLs are modal logics developed to specify and reason about program schemas in terms of states and state transitions caused by (running) a program (Fischer & Ladner, 1979; Kozen & Tiuryn, 1990).

The correspondence between DLs and PDLs is due to a substantial similarity between the interpretation structure of the two kinds of logics: individuals in DLs correspond to states in PDLs, links between individuals correspond to state transitions, concepts correspond to formulae, and roles correspond to programs. In fact, most constructs in DLs have a counterpart in known PDLs as shown in Figure 3. In (Schild, 1991), using the correspondence, many new results on DLs were obtained from known results on PDLs. In particular, from the decision procedures for Converse PDL, the first reasoning procedures for DLs that include inverse and TBoxes were devised.

Notably, neither qualified number restrictions nor ABoxes have a counterpart in PDLs.

Indeed the only form of number restrictions known in PDLs is that of assuming all atomic programs (not

DLs		PDLs	
atomic concept	A	atomic proposition	A
top	$\top$	true	tt
bottom	$\bot$	false	ff
conjunction	$C_1 \sqcap C_2$	conjunction	$\phi_1 \wedge \phi_2$
disjunction	$C_1 \sqcup C_2$	disjunction	$\phi_1 \vee \phi_2$
negation	$\neg C$	negation	$\neg\phi$
existential quantification	$\exists R.C$	diamond ("some runs ...")	$<r>\phi$
universal quantification	$\forall R.C$	box ("all runs ...")	$[r]\phi$
qualified number restrictions	$(\geq n\, Q.C)$ $(\leq n\, Q.C)$	(deterministic PDLs)	**assumption:** deterministic atomic programs
atomic role	P	atomic program	P
union	$R_1 \sqcup R_2$	choice	$r_1 \cup r_2$
chaining	$R_1 \circ R_2$	sequence	$r_1; r_2$
reflexive-transitive closure	R^*	reflexive-transitive closure	r^*
test	$id(C)$	test	$\phi?$
inverse	R^-	converse	r^-
basic role	Q	—	—
inclusion assertions	$C_1 \sqsubseteq C_2$	axioms (valid formulae)	ϕ
instance assertions	$C(\alpha) \mid P(\alpha_1, \alpha_2)$	—	—

Figure 3: Correspondence between DLs and PDLs.

their inverse) to be deterministic, thus getting the so call Deterministic PDLs. As an aside, Deterministic PDLs that include also the converse operator do not have the finite model property and indeed the reasoning procedures developed for these logics are based on the construction of automata on infinite trees.

As for ABoxes, in PDLs, they would roughly correspond to a partial specification of an actual evaluation of a program. However, such kind of specification have not been studied yet.

The research in (De Giacomo & Lenzerini, 1994a, 1995; De Giacomo, 1995) has tackled these two aspects.

In (De Giacomo & Lenzerini, 1994a) the EXPTIME-decidability of $\mathcal{CIF}$, i.e. $\mathcal{CIQ}$ with number restrictions limited to unqualified functional restrictions (on both atomic roles and their inverse), was established. In (De Giacomo & Lenzerini, 1995; De Giacomo, 1995) this result was extended to $\mathcal{CIQ}$. The reasoning procedures developed in these works do not construct automata on infinite trees, but are based on a polynomial encoding of a $\mathcal{CIQ}$ TBox into a $\mathcal{CIF}$ TBox, which is in turn encoded into a $\mathcal{CI}$-concept (corresponding to a Converse PDL formula. Observe that, from $\mathcal{CIQ}$ to $\mathcal{CI}$, we go from a logic which does not have the finite model property to a logic that does have it.

As for ABoxes, the best known results about reasoning on knowledge bases constituted by both a TBox and an ABox, are two EXPTIME reasoning procedures presented in (De Giacomo & Lenzerini, 1994a; De Giacomo, 1995) for $\mathcal{CI}$ and $\mathcal{CQ}$ (the logic obtainded from $\mathcal{CIQ}$ by disallowing inverse roles), respectively. Notice that both logics have the finite model property.

Finally we remark that in $\mathcal{CIQ}$, qualified number restrictions are allowed only for basic roles (i.e. atomic roles and their inverse). This is a disign choice due to the fact that allowing a generic role to appear in a qualified number restriction would have made the logic undecidable. Indeed it suffices to observe that the unqualified functional restriction $(\leq 1\,(R_1 \sqcup R_2).\top)$ is in fact a form of role value map[1], which leads to undecidability (e.g. see (Schmidt-Schauß, 1989)).

5 TECHNICAL PRELIMINARIES

We assume, without loss of generality, $\sqcup, \forall, \leq$ to be expressed by means of $\neg, \sqcap, \exists, \geq$, and the inverse role operator to be applied to atomic roles only[2].

The *Fisher-Ladner closure* (Fischer & Ladner, 1979) of a $\mathcal{CIQ}$ concept C is denoted by $CL(C)$ and is defined

[1]This observation is originally due to Franz Baader.

[2]We recall that the following equations hold: $(R_1 \circ R_2)^- = R_2^- \circ R_1^-$, $(R_1 \sqcup R_2)^- = R_1^- \sqcup R_2^-$, $(R_1^*)^- = (R_1^-)^*$, $id(C)^- = id(C)$.

inductively as the smallest set of concepts S containing C and such that:

$$
\begin{array}{ll}
C_1 \sqcap C_2 \in S & \text{implies } C_1, C_2 \in S \\
\neg C' \in S & \text{implies } C' \in S \\
C' \in S & \text{implies } \neg C' \in S \text{ (if } C' \neq \neg C'') \\
(\geq n\, Q.C') \in S & \text{implies } C' \in S \\
\exists R.C' \in S & \text{implies } C' \in S \\
\exists R_1 \circ R_2.C' \in S & \text{implies } \exists R_1.\exists R_2.C' \in S \\
\exists R_1 \sqcup R_2.C' \in S & \text{implies } \exists R_1.C', \exists R_2.C' \in S \\
\exists R^*.C' \in S & \text{implies } \exists R.\exists R^*.C' \in S \\
\exists id(C'').C' \in S & \text{implies } C'' \in S.
\end{array}
$$

Intuitively, $CL(C)$ is analogous to the set of subconcepts in simpler logics: It comprises the concepts that play a direct role in establishing the interpretation of C. The size of $CL(C)$ is linearly bounded by the size of C (cf. (Fischer & Ladner, 1979)). By definition, if $C' \in CL(C)$, then $CL(C') \subseteq CL(C)$.

We can extend the above notion to that of the Fisher-Ladner closure of a knowledge base by simply taking the union of the Fisher-Ladner closures of all concepts appearing in the knowledge base.

Let us denote the *empty sequence of roles* by the role ε, and define $\exists\varepsilon.C \doteq C$ and $\forall\varepsilon.C \doteq C$. Given a role R, we call $Pre(R)$ and $Post(R)$ the two sets of roles defined inductively as follows ($Q = P \mid P^-$):

$$
\begin{array}{ll}
Pre(Q) & = \{\varepsilon, Q\} \\
Pre(R_1 \circ R_2) & = \{R_1 \circ R_2' \mid R_2' \in Pre(R_2)\} \cup Pre(R_1) \\
Pre(R_1 \sqcup R_2) & = Pre(R_1) \cup Pre(R_2) \\
Pre(R_1^*) & = \{R_1^* \circ R_1' \mid R_1' \in Pre(R_1)\} \\
Pre(id(C)) & = \{\varepsilon, id(C)\}
\end{array}
$$

$$
\begin{array}{ll}
Post(Q) & = \{\varepsilon, Q\} \\
Post(R_1 \circ R_2) & = \{R_1' \circ R_2 \mid R_1' \in Post(R_1)\} \cup Post(R_2) \\
Post(R_1 \sqcup R_2) & = Post(R_1) \cup Post(R_2) \\
Post(R_1^*) & = \{R_1' \circ R_1^* \mid R_1' \in Post(R_1)\} \\
Post(id(C)) & = \{\varepsilon, id(C)\}.
\end{array}
$$

Roughly speaking, $Pre(R)$ and $Post(R)$ are the sets formed by those roles that are "prefix" and "postfix" of the role R, respectively. The size of both $Pre(R)$ and $Post(R)$ is polynomial in the size of R.

For the roles in $Post(R)$ the following two properties can be easily proven (see (De Giacomo, 1996, 1995)):

- Let $\exists R.C$ be a concept. For all roles $R' \in Post(R)$, $\exists R'.C \in CL(\exists R.C)$.
- Let $\exists R_1.\ldots.\exists R_l.C$ be a concept. For all roles $R' \in Post(R_1 \circ \ldots \circ R_l)$, there is a formula $D \in CL(\exists R_1.\ldots.\exists R_l.C)$ such that D is equivalent to $\exists R'.C$.

A *path* in an interpretation $\mathcal{I}$ is a sequence $(s_0, \ldots, s_q)$ of elements of $\Delta^{\mathcal{I}}$ ($q \geq 0$), such that for each $i = 1, \ldots, q$, $(s_{i-1}, s_i) \in Q^{\mathcal{I}}$, for some $Q = P \mid P^-$. The length of $(s_0, \ldots, s_q)$ is q. Intuitively a path describes the sequence of individuals which are met by following a role (or the inverse of a role) in a given interpretation. We inductively define the set of paths $Paths_{\mathcal{I}}(R)$ of a role R in an interpretation $\mathcal{I}$, as follows, where ($Q = P \mid P^-$):

$$
\begin{array}{ll}
Paths_{\mathcal{I}}(Q) & = Q^{\mathcal{I}} \\
Paths_{\mathcal{I}}(R_1 \sqcup R_2) & = Paths_{\mathcal{I}}(R_1) \cup Paths_{\mathcal{I}}(R_2) \\
Paths_{\mathcal{I}}(R_1 \circ R_2) & = \{(s_0, \ldots, s_u, \ldots, s_q) \mid \\
 & \quad (s_0, \ldots, s_u) \in Paths_{\mathcal{I}}(R_1) \text{ and} \\
 & \quad (s_u, \ldots, s_q) \in Paths_{\mathcal{I}}(R_2)\} \\
Paths_{\mathcal{I}}(R^*) & = \{(s) \mid s \in \Delta^{\mathcal{I}}\} \cup (\bigcup_{i>0} Paths_{\mathcal{I}}(R^i)) \\
Paths_{\mathcal{I}}(id(C)) & = \{(s) \mid s \in C^{\mathcal{I}}\}.
\end{array}
$$

We say that a path (s_0) in $\mathcal{I}$ *satisfies* a concept C which is not of the form $\exists R.C$ if $s_0 \in C^{\mathcal{I}}$. We say that a path $(s_0, \ldots, s_q)$ in $\mathcal{I}$ *satisfies* a concept C of the form $\exists R_1.\cdots\exists R_l.C'$, where C' is not of the form $\exists R'.C''$, if $(s_0, \ldots s_q) \in Paths_{\mathcal{I}}(R_1 \circ \cdots \circ R_l)$ and $s_q \in C'^{\mathcal{I}}$.

The following two propositions describe the basic properties of paths and can be proven by induction on the structure of the role R (see (De Giacomo, 1996, 1995)).

Proposition 1 *Let $\mathcal{I}$ be an interpretation and $\exists R.C$ a concept such that: $s \in (\exists R.C)^{\mathcal{I}}$, $(s) \in Paths_{\mathcal{I}}(R)$, and $s \in C^{\mathcal{I}}$. Then there exists a concept $\exists id(C_1) \circ \ldots \circ id(C_g).C$, with $g \geq 0$, such that:*

- *all tests $id(C_i)$ occur in R, and hence $C_i \in CL(\exists R.C)$;*
- *$s \in (\exists id(C_1) \circ \ldots \circ id(C_g).C)^{\mathcal{I}}$;*
- *$\exists id(C_1) \circ \ldots \circ id(C_g).C \sqsubseteq \exists R.C$ is valid.*

Proposition 2 *Let $\mathcal{I}$ be a structure, and $\exists R.C$ a formula such that: $s \in (\exists R.C)^{\mathcal{I}}$, $(s = s_0, \ldots, s_q) \in Paths_{\mathcal{I}}(R)$ with $q > 0$, $s_q \in C^{\mathcal{I}}$. Then there exists a formula $\exists id(C_1) \circ \ldots \circ id(C_g) \circ Q.\exists R'.C$, with $g \geq 0$, such that:*

- *all tests $id(C_i)$ occur in R, and hence $C_i \in CL(\exists R.C)$;*
- *$R' \in Post(R)$ and hence $\exists R'.C$ is equivalent to D for some $D \in CL(\exists R.C)$;*

- $(s_0, s_1) \in Q^{\mathcal{I}}$;
- $s_1 \in (\exists R'.C)^{\mathcal{I}}$;
- $(s_1, \ldots, s_q) \in Paths_{\mathcal{I}}(R')$;
- $\exists id(C_1) \circ \ldots \circ id(C_g) \circ Q.(\exists R'.C) \sqsubseteq \exists R.C$ *is valid.*

6 REASONING IN $\mathcal{CIQ}$ KNOWLEDGE BASES

In this section, we illustrate the technique for reasoning on $\mathcal{CIQ}$ knowledge bases. The basic idea underlying our method is as follows: checking the satisfiability of a $\mathcal{CIQ}$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is polynomially reduced to checking the satisfiability of a $\mathcal{CIQ}$ knowledge base $\mathcal{K}' = (\mathcal{T}', \mathcal{A}')$, whose ABox $\mathcal{A}'$ is made of a single instance assertion $C(\alpha)$. In other words, the satisfiability of $\mathcal{K}$ is reduced to the satisfiability of the concept C wrt the TBox $\mathcal{T}'$ of the resulting knowledge base. The latter reasoning service can be realized by means of the method presented in (De Giacomo & Lenzerini, 1995; De Giacomo, 1995), and is known to be EXPTIME-complete. Thus, by means of the reduction, we get an EXPTIME algorithm for satisfiability of $\mathcal{CIQ}$ knowledge base, and hence for all reasoning services on $\mathcal{CIQ}$ knowledge bases.

Definition Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a $\mathcal{CIQ}$ knowledge base. We call *the reduced form* of $\mathcal{K}$ the $\mathcal{CIQ}$ knowledge base $\mathcal{K}' = (\mathcal{T}', \mathcal{A}')$ defined as follows (a new atomic concept A_i is introduced for each individual α_i (i=1,...,m) occurring in $\mathcal{A}$).

- $\mathcal{A}' = \{(\exists create.A_1 \sqcap \ldots \sqcap \exists create.A_m)(g)\}$, where g is a new individual (the only one present in $\mathcal{A}'$) and *create* is a new atomic role:
- $\mathcal{T}'$ is formed by $\mathcal{T}'_{\mathcal{K}}$ and $\mathcal{T}'_{aux}$:
  - $\mathcal{T}'_{\mathcal{K}} = \mathcal{T}'_{\mathcal{T}} \cup \mathcal{T}'_{\mathcal{A}}$, where $\mathcal{T}'_{\mathcal{T}} = \mathcal{T}$, and $\mathcal{T}'_{\mathcal{A}}$ is made of one inclusion assertion:

    $$A_i \sqsubseteq C$$

    for each instance assertion $C(\alpha_i) \in \mathcal{A}$, two inclusion assertions:

    $$A_i \sqsubseteq \exists P.A_j \sqcap (\leq 1\, P.A_j)$$
    $$A_j \sqsubseteq \exists P^-.A_i \sqcap (\leq 1\, P^-.A_i)$$

    for each instance assertion $P(\alpha_i, \alpha_j) \in \mathcal{A}$, and one inclusion assertion:

    $$A_i \sqsubseteq \sqcap_{i \neq j} \neg A_j$$

    for each individual α_i occurring in $\mathcal{A}$.
  - $\mathcal{T}'_{aux}$ is made of one inclusion assertion (**u** stands for $(P_1 \sqcup \ldots \sqcup P_n \sqcup P_1^- \sqcup \ldots \sqcup P_n^-)^*$, where $P_1, \ldots, P_n$ are all the atomic roles in $\mathcal{T}'_{\mathcal{K}}$):

    $$(A_i \sqcap C) \sqsubseteq \forall \mathbf{u}.(\neg A_i \sqcup C)$$

    for each A_i occurring in $\mathcal{T}'_{\mathcal{K}}$ and C such that:
    1. $C \in CL(\mathcal{T}'_{\mathcal{K}})$
    2. $C = \exists \overline{R}.C'$ with $\exists R.C' \in CL(\mathcal{T}'_{\mathcal{K}})$
    3. $C = \exists(\overline{R'} \circ Q).A_j$ with $R' \in Pre(R)$, $Q = P \mid P^-$, and R, P, A_j occurring in $CL(\mathcal{T}'_{\mathcal{K}})$

    where, $\overline{R}$ is defined inductively as follows ($Q = P \mid P^-$):
    - $\overline{Q} = Q \circ id(\sqcap_i \neg A_i)$;
    - $\overline{R_1 \circ R_2} = \overline{R_1} \circ \overline{R_2}$;
    - $\overline{R_1 \sqcup R_2} = \overline{R_1} \sqcup \overline{R_2}$;
    - $\overline{R_1^*} = \overline{R_1}^*$;
    - $\overline{id(C)} = id(C)$.

□

Lemma 3 *Let $\mathcal{K}$ be a $\mathcal{CIQ}$ knowledge base, and $\mathcal{K}'$ its reduced form. Then the size of $\mathcal{K}'$ is polynomial with respect to the size of $\mathcal{K}$.*

Let us comment on how the reduced form $\mathcal{K}' = (\mathcal{T}', \mathcal{A}')$ relates to the original knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. First, observe that the ABox $\mathcal{A}'$ is used to force the existence of the only individual g, connected by the role *create* to one instance of each A_i. It can be shown that this allows us to restrict the attention to models of $\mathcal{K}'$ that represent graphs connected to g, i.e. models $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of $\mathcal{K}'$ such that $\Delta^{\mathcal{I}} = \{g\} \cup \{s' \mid (g, s') \in create^{\mathcal{I}} \circ (\bigcup_P (P^{\mathcal{I}} \sqcup P^{\mathcal{I}^-})^*$.

The TBox $\mathcal{T}'$ consists of two parts $\mathcal{T}'_{\mathcal{K}}$ and $\mathcal{T}'_{aux}$. $\mathcal{T}'_{\mathcal{K}}$ is made of the original inclusion assertions in $\mathcal{T}$ plus what we may call a "naive encoding" of the original ABox $\mathcal{A}$ as inclusion assertions, which form $\mathcal{T}'_{\mathcal{A}}$. Indeed, each individual α_i is represented in $\mathcal{T}'_{\mathcal{A}}$ as a new atomic concept A_i (disjoint from the other A_j's), and the instance assertions in the original ABox $\mathcal{A}$ are represented as inclusion assertions in $\mathcal{T}'_{\mathcal{A}}$ involving such new atomic concepts. However $\mathcal{T}'_{\mathcal{K}}$ alone does not suffice to represent faithfully (wrt the reasoning services we are interested in) the original knowledge base, because an individual α_i in $\mathcal{K}$ is represented by the set of instances of A_i in $\mathcal{K}'$. In order to relate the satisfiability of $\mathcal{K}'$ to the satisfiability of $\mathcal{K}$, we must be able to single out, for each A_i, one instance of A_i representative of α_i. For this purpose, we need to add a new part, called $\mathcal{T}'_{aux}$, to $\mathcal{T}'$. Roughly speaking, $\mathcal{T}'_{aux}$ contains inclusion assertions of the form:

$$(A_i \sqcap C) \sqsubseteq \forall \mathbf{u}.(\neg A_i \sqcup C)$$

which say that if an instance of A_i is also an instance of C,a new then every instance of A_i is an instance of C. Observe that if we could add an infinite set of assertions of this form, one for each possible concept of the language (e.g. by a kind of axiom schema), we could safely restrict our attention to models of $\mathcal{K}'$ with just one instance for every concept A_i $(i = 1, \ldots, m)$, since there would be no way in the logic to distinguish two instances of A_i one from the other. What we show below is that in fact we do need only a finite (polynomial) number of such inclusion assertions (as specified by T'_{aux}) in order to be able to identify, for each i, an instance of A_i as representative of α_i. This allows us to prove that the existence of a model of $\mathcal{K}'$ implies the existence of a model of $\mathcal{K}$.

The individuals t of a model $\mathcal{I}$ of $\mathcal{K}'$ such that $t \in A_i^{\mathcal{I}}$ are called *aliases* of the individual α_i in $\mathcal{I}$. The assertions in $\mathcal{T}'_{aux}$ allow us to prove the lemma below.

Lemma 4 *Let $\mathcal{K}$ be a $\mathcal{CIQ}$ knowledge base, $\mathcal{K}'$ its reduced form, and $\mathcal{I}$ a model of $\mathcal{K}'$. Let t be an alias of α_i in $\mathcal{I}$, and let $\exists R.C \in CL(\mathcal{T}'_{\mathcal{K}})$. If there is a path from t that satisfies $\exists R.C$ and contains N aliases $t = t_1, \ldots, t_N$, of $\alpha_i = \alpha_{i_1}, \ldots, \alpha_{i_N}$ respectively, then from every alias t' of α_i in $\mathcal{I}$, there is a path that satisfies $\exists R.C$ and contains N aliases $t' = t'_1, \ldots, t'_N$ of $\alpha_{i_1}, \ldots, \alpha_{i_N}$, in the same order as $t_1, \ldots, t_N$.*

Proof By induction on the number N of aliases, making use of the inclusion assertions in $\mathcal{T}'_{aux}$ with C of the form (2) and (3). □

We further restrict our attention to tree-like models only, without loss of generality. Indeed, any model $\mathcal{I}$ of $\mathcal{K}'$ can be easily transformed into a tree-like model, by simply unfolding $\mathcal{I}$ as follows: Put g as the root of the tree; for each Q-successor $(Q = P \mid P^-)$ of g add it to the tree as a Q-child of the node; continue recursively to process the children of g, and so on. Observe that the tree-like model obtained may be infinite.

Note that, by virtue of $\mathcal{A}'$, in the tree-like model, for each α_i occurring in $\mathcal{K}$, g has one *create*-successor s_{α_i}, as a child, such that $s_{\alpha_i} \in A_i$. Moreover, each s_{α_i} has a single P-successor $s \in A_j^{\mathcal{I}}$ for each $P(\alpha_i, \alpha_j) \in \mathcal{K}$ and a single P-predecessor $s' \in A_j^{\mathcal{I}}$ for each $P(\alpha_j, \alpha_i) \in \mathcal{K}$, by $\mathcal{T}'_{\mathcal{A}}$.

Given a tree-like model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of $\mathcal{K}'$, we define a new interpretation $\mathcal{I}' = (\Delta^{\mathcal{I}'}, \cdot^{\mathcal{I}'})$ of $\mathcal{K}'$ as follows

- $\Delta^{\mathcal{I}'} = \{g\} \cup \{s \in \Delta^{\mathcal{I}} \mid (g, s) \in \mathcal{R}_{create} \circ (\bigcup_P (\mathcal{R}_P \cup \mathcal{R}_P^-))^*\}$
- $create^{\mathcal{I}} = \mathcal{R}_{create}$ and $P^{\mathcal{I}} = \mathcal{R}_P \cap (\Delta^{\mathcal{I}'} \times \Delta^{\mathcal{I}'})$ for each atomic role P occurring in $\mathcal{K}'$
- $A^{\mathcal{I}'} = A^{\mathcal{I}} \cap \Delta^{\mathcal{I}'}$ for each atomic concept A occurring in $\mathcal{K}'$

where

- $\mathcal{R}_{create} = \{(g, s_{\alpha_i}) \in create^{\mathcal{I}} \mid \alpha_i \text{ for } i = 1, \ldots, m\}$
- $\mathcal{R}_P = (P^{\mathcal{I}} - (\{(s_{\alpha_i}, s) \in P^{\mathcal{I}} \mid s \in A_j^{\mathcal{I}} \text{ and } P(\alpha_i, \alpha_j) \in \mathcal{K}\} \cup \{(s_{\alpha_j}, s') \in (P^-)^{\mathcal{I}} \mid s' \in A_i^{\mathcal{I}} \text{ and } P(\alpha_i, \alpha_j) \in \mathcal{K}\})) \cup \{(s_{\alpha_i}, s_{\alpha_j}) \mid P(\alpha_i, \alpha_j) \in \mathcal{K}\}$.

Observe that in $\mathcal{I}'$, for every atomic role P, the number of P-successors of all individuals in $\Delta^{\mathcal{I}'}$, is the same as in $\mathcal{I}$. The following lemma holds for $\mathcal{I}'$.

Lemma 5 *Let $\mathcal{K}$ be a $\mathcal{CIQ}$ knowledge base and $\mathcal{K}'$ its reduced form. Let $\mathcal{I}$ be a model of $\mathcal{K}' = (\mathcal{T}', \mathcal{A}')$, and $\mathcal{I}'$ be the interpretation obtained from $\mathcal{I}$ as above. Then, for every $C \in CL(\mathcal{T}'_{\mathcal{K}})$ and for every $x \in \Delta^{\mathcal{I}'}$:*

$$x \in C^{\mathcal{I}'} \text{ if and only if } x \in C^{\mathcal{I}}.$$

Proof By induction on the formation of C (called concept induction in the following). The only complex case is $C = \exists R.C'$. Here we show the if-direction of such a case (the only-if-direction is similar, yet slightly simpler).

If $x \in (\exists R.C')^{\mathcal{I}}$, then there is a path $(x = x_0, \ldots, x_q) \in Paths_{\mathcal{I}}(R)$ such that $x_q \in C'^{\mathcal{I}}$. We prove $x \in (\exists R.C')^{\mathcal{I}'}$, by induction on the number k of aliases along the path $(x_0, \ldots, x_q)$, different from s_{α_i} for any i (we call this induction, path induction).

Case $k = 0$. In this case, for all the states x_i along the path, $x_i \in \Delta^{\mathcal{I}'}$. By applying Proposition 2 q times and Proposition 1 once, we can conclude that there exists a concept $\exists((id(C_{0,1}) \circ \ldots \circ id(C_{0,g_0})) \circ Q_1 \circ \ldots \circ (id(C_{q-1,1}) \circ \ldots \circ id(C_{q-1,g_{q-1}})) \circ Q_q \circ (id(C_{q,1}) \circ \ldots \circ id(C_{q,g_q}))).C'$ with $g_i \geq 0$, such that:

- all tests $id(C_{i,j})$ occur in R, and hence $C_{i,j} \in CL(\exists R.C') \subseteq CL(\mathcal{T}'_{\mathcal{K}})$;
- $(x_{i-1}, x_i) \in Q_i^{\mathcal{I}}$, for $i = 1, \ldots, q$;
- $\exists((id(C_{0,1}) \circ \ldots \circ id(C_{0,g_0})) \circ Q_1 \circ \ldots \circ (id(C_{q-1,1}) \circ \ldots \circ id(C_{q-1,g_{q-1}})) \circ Q_q \circ (id(C_{q,1}) \circ \ldots \circ id(C_{q,g_q}))).C' \sqsubseteq \exists R.C'$ is valid.

By concept induction hypothesis we have that, for all $C_{i,j}$, $x_i \in C_{i,j}^{\mathcal{I}}$ iff $x_i \in C_{i,j}^{\mathcal{I}'}$, and $x_q \in C'^{\mathcal{I}}$ iff $x_q \in C'^{\mathcal{I}'}$. By construction of $\mathcal{I}'$, $(x_{i-1}, x_i) \in Q_i^{\mathcal{I}}$ implies $(x_{i-1}, x_i) \in Q_i^{\mathcal{I}'}$. Hence $x \in (\exists R.C')^{\mathcal{I}'}$.

Case $k > 0$. Let $(x_0, \ldots, x_q) = (x_0, \ldots, x_u, \ldots x_q)$ where x_u, such that $x_u \in A_j^{\mathcal{I}}$, is the first alias, different from s_{α_i} for any i, along the path $(x_0, \ldots, x_q)$. By applying Proposition 2 u times only, we can conclude that, there exists a formula $\exists((id(C_{0,1}) \circ \ldots \circ id(C_{0,g_0})) \circ Q_1 \circ \ldots \circ (id(C_{u-1,1}) \circ \ldots \circ id(C_{q-1,g_{u-1}})) \circ Q_u).(\exists R'.C')$ with $g_i \geq 0$, such that:

- all tests $id(C_{i,j})$ occur in R, and hence $C_{i,j} \in CL(\exists R.C') \subseteq CL(\mathcal{T}'_{\mathcal{K}})$;
- $R' \in Post(R)$, and hence the concept $\exists R'.C'$ is equivalent to D for some $D \in CL(\exists R.C') \subseteq CL(\mathcal{T}'_{\mathcal{K}})$;
- $(x_{i-1}, x_i) \in Q_i^{\mathcal{I}}$, for $i = 1, \ldots u$;
- $(x_u, \ldots, x_q) \in Paths_{\mathcal{I}}(R')$;
- $\exists((id(C_{0,1}) \circ \ldots id(C_{0,g_0})) \circ Q_1 \circ \ldots \circ (id(C_{q-1,1}) \circ \ldots \circ id(C_{q-1,g_{q-1}})) \circ Q_q).(\exists R'.C') \sqsubseteq \exists R.C'$ is valid.

Since the path $(x_u, \ldots, x_q)$ contains k aliases, by Lemma 4, from each alias of α_j there is a path satisfying $\exists R'.C'$ which goes through exactly the "same" k aliases in the same order. Let $(s_{\alpha_i} = x'_u, \ldots, x'_{q'})$ be such a path. This path contains less than k aliases, excluding x'_u. Thus, by path induction hypothesis, $s_{\alpha_i} \in (\exists R'.C')^{\mathcal{I}'}$.

Now, by construction of $\mathcal{I}'$, $(x_{u-1}, x_u) \in Q_u^{\mathcal{I}}$ implies $(x_{u-1}, s_{\alpha_i}) \in Q_u^{\mathcal{I}'}$ thus $x_{u-1} \in (\exists Q_u.(\exists R'.C'))^{\mathcal{I}'}$. Whereas, by formula induction hypothesis, for all $C_{i,j}$, $x_i \in C_{i,j}^{\mathcal{I}}$ iff $x_i \in C_{i,j}^{\mathcal{I}'}$. Hence considering that for $i = 1, \ldots, u-1$, $(x_{i-1}, x_i) \in Q_i^{\mathcal{I}}$ implies $(x_{i-1}, x_i) \in Q_i^{\mathcal{I}'}$, we get $x \in (\exists R.C')^{\mathcal{I}'}$. □

We can now state the main theorem on reasoning in $\mathcal{CIQ}$ knowledge bases.

Theorem 6 *A $\mathcal{CIQ}$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is satisfiable iff its reduced form $\mathcal{K}' = (\mathcal{T}', \mathcal{A}')$ is satisfiable. Thus, satisfiability of $\mathcal{CIQ}$ knowledge bases is EXPTIME-complete.*

Proof ⇒ We can extend a model $\mathcal{I}$ of $\mathcal{K}$ to a model of $\mathcal{K}'$ by adding the individual g to $\Delta^{\mathcal{I}}$, and letting $create^{\mathcal{I}} = \{(g, \alpha_i) \mid i = 1, \ldots, m\}$.

⇐ If there exists a model $\mathcal{I}$ of $\mathcal{K}'$ then by Lemma 5 we can construct an interpretation $\mathcal{I}'$ such that (1) $\mathcal{I}'$ satisfies all inclusion assertions in $\mathcal{T}$; (2) to each individual α_i occurring in $\mathcal{K}$, it corresponds exactly one individual s_{α_i} of $\mathcal{I}'$, and for such s_{α_i} we have $s_{\alpha_i} \in C^{\mathcal{I}}$ for each instance assertion $C(\alpha_i)$ in $\mathcal{K}$, and $(\alpha_i, \alpha_j) \in P^{\mathcal{I}'}$ for each instance assertion $P(\alpha_i, \alpha_j)$. Hence $\mathcal{I}'$ satisfies $\mathcal{K}$.

Thus, the satisfiability of $\mathcal{K}$ is polynomially reducible to satisfiability of its reduced form $\mathcal{K}'$, i.e. to satisfiability of a concept (namely the concept in $\mathcal{A}'$) in a $\mathcal{CIQ}$ TBox (namely $\mathcal{T}'$), which is known to be EXPTIME-complete (De Giacomo & Lenzerini, 1995; De Giacomo, 1995). □

7 DISCUSSION AND CONCLUSION

We have presented a new technique for reasoning in a DL system with full capabilities, showing that reasoning in this logic is EXPTIME-complete. The technique is based on a careful encoding of instance assertions into special TBox assertions, that treat individuals as mutually disjoint atomic concepts, and add suitable constraints by exploiting the capability of $\mathcal{CIQ}$ to express complex properties of role chains. We stress the importance of these additional constraints: indeed, a naive translation of individuals into atomic concepts, like the one implicitly done in CLASSIC (Borgida & Patel-Schneider, 1994), would not be sufficient for our purposes.

Consider the knowledge base $\mathcal{K}$ constituted by the TBox and the ABox shown in Figure 4.

The first inclusion assertion constrains the role succ and its inverse to be functionl.

The second inclusion assertion states that every individual is linked by a (succ ⊔ succ⁻)-chain to some instance of C. In fact the existence of a single instance of C for each (succ ⊔ succ⁻)-connected part of the model is sufficient to satisfy the above constraint.

The third inclusion assertion states that the instances of C have a succ-successor in D and a succ-predecessor in ¬D.

The assertions in the ABox express that a has b as succ-successor, and b has a as succ-successor.

The knowledge base $\mathcal{K}$ is unsatisfiable. Indeed, both a and b must be connected by a (succ ⊔ succ⁻)-chain to an instance of C, hence either a or b must be an instance of C. Suppose that b is an instance of C. Then its succ-successor, which is a, is an instance of D, and

TBox:	ABox:	naive ABox encoding
$\top \sqsubseteq (\leq 1\ \mathtt{succ}.\top) \sqcap (\leq 1\ \mathtt{succ}^-.\top)$ $\top \sqsubseteq \exists(\mathtt{succ} \sqcup \mathtt{succ}^-)^*.\mathtt{C}$ $\mathtt{C} \sqsubseteq \exists \mathtt{succ}.\mathtt{D} \sqcap \exists \mathtt{succ}.\neg \mathtt{D}$	succ(a, b) succ(b, a)	$\mathtt{A} \sqsubseteq \exists \mathtt{succ}.\mathtt{B}$ $\mathtt{B} \sqsubseteq \exists \mathtt{succ}^-.\mathtt{A}$ $\mathtt{B} \sqsubseteq \exists \mathtt{succ}.\mathtt{A}$ $\mathtt{A} \sqsubseteq \exists \mathtt{succ}^-.\mathtt{B}$

Figure 4: Example: naive ABox encoding fails.

its succ-predecessor, which is again a, is an instance of $\neg$D. But this is a contradiction. The conclusion is reached if we assume that a is an instance of C.

Now consider the knowledge base $\mathcal{K}'$ obtained from $\mathcal{K}$ by substituting the assertions in the ABox with their naive encodings (see Fig. 4, where A and B are disjoint). It is easy to see that $\mathcal{K}'$ is satisfiable. Indeed, to see this, it suffices to consider the following interpretation: $\Delta^{\mathcal{I}} = \mathtt{a}, \mathtt{a}', \mathtt{b}, \mathtt{b}'$, $\mathtt{a}, \mathtt{a}' \in \mathtt{A}^{\mathcal{I}}$, $\mathtt{b}, \mathtt{b}' \in \mathtt{B}^{\mathcal{I}}$, $(\mathtt{a}, \mathtt{b}), (\mathtt{b}, \mathtt{a}'), (\mathtt{a}', \mathtt{b}'), (\mathtt{b}', \mathtt{a}) \in \mathtt{succ}^{\mathcal{I}}$, and such that $\mathtt{b} \in \mathtt{C}^{\mathcal{I}}$, $\mathtt{a} \in \mathtt{D}^{\mathcal{I}}$ and $\mathtt{a}' \in \neg \mathtt{D}^{\mathcal{I}}$.

Observe that if we include the assertions:

$$(\mathtt{A} \sqcap \mathtt{D}) \sqsubseteq \forall \mathtt{u}.(\neg \mathtt{A} \sqcup \mathtt{D})$$

for all $\mathtt{D} \in CL(\mathcal{K})$, then the above interpretation is not a model anymore.

By virtue of the characteristics of the encoding presented in this paper, it can be shown that the technique can be extended to even more powerful DLs, such as $\mathcal{CATS}$ and $\mathcal{CVL}$ (see (De Giacomo & Lenzerini, 1995; Calvanese et al., 1995)), which include role conjunction and a limited form of role-value map. Here, we restricted our attention to $\mathcal{CIQ}$ for the sake of simplicity.

In the future, we aim at extending our analysis to the ONE-OF construct (by which we can form a concept as a set of individuals). Although we know that reasoning is still EXPTIME decidable if we add ONE-OF and get rid off of either inverse roles or number restrictions (De Giacomo & Lenzerini, 1994a; De Giacomo, 1995), the decidability of reasoning on $\mathcal{CIQ}$ knowledge bases extended with ONE-OF is still an open problem.

References

Artale, A., & Franconi, E. (1994). A computational account for a description logic of time and action. In Doyle, J., Sandewall, E., & Torasso, P. (Eds.), *Proc. of KR-94*, pp. 3–14 Bonn. Morgan Kaufmann, Los Altos.

Baader, F. (1991). Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In *Proc. of IJCAI-91* Sydney, Australia.

Baader, F., Hollunder, B., Nebel, B., Profitlich, H.-J., & Franconi, E. (1992). An empirical analysis of optimization techniques for terminological representation systems. In *Proc. of KR-92*, pp. 270–281. Morgan Kaufmann, Los Altos.

Bergamaschi, S., & Sartori, C. (1992). On taxonomic reasoning in conceptual design. *ACM Trans. on Database Systems*, *17*(3), 385–422.

Blanco, J. M., Illarramendi, A., & Goni, A. (1994). Building a federated database system: An approach using a knowledge based system. *J. of Intelligent and Cooperative Information Systems*, *3*(4), 415–455.

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

Borgida, A., & Patel-Schneider, P. F. (1994). A semantics and complete algorithm for subsumption in the CLASSIC description logic. *J. of Artificial Intelligence Research*, *1*, 277–308.

Buchheit, M., Donini, F. M., & Schaerf, A. (1993). Decidable reasoning in terminological knowledge representation systems. *J. of Artificial Intelligence Research*, *1*, 109–138.

Calvanese, D., De Giacomo, G., & Lenzerini, M. (1995). Structured objects: Modeling and reasoning. In *Proc. of DOOD-95*, No. 1013 in LNCS, pp. 229–246. Springer-Verlag.

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

De Giacomo, G. (1995). *Decidability of Class-Based Knowledge Representation Formalisms*. Ph.D. thesis, Dip. di Inf. e Sist., Univ. di Roma "La Sapienza".

De Giacomo, G. (1996). Eliminating converse from Converse PDL. *J. of Logic, Language and Information*. To appear.

De Giacomo, G., & Lenzerini, M. (1994a). Boosting the correspondence between description logics and propositional dynamic logics. In *Proc. of AAAI-94*, pp. 205–212. AAAI Press/The MIT Press.

De Giacomo, G., & Lenzerini, M. (1994b). Concept language with number restrictions and fixpoints, and its relationship with μ-calculus. In *Proc. of ECAI-94*, pp. 411–415.

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

Devambu, P., Brachman, R. J., Selfridge, P. J., & Ballard, B. W. (1991). LASSIE: A knowledge-based software information system. *Comm. of the ACM*, *34*(5), 36–49.

Donini, F. M., Lenzerini, M., Nardi, D., & Nutt, W. (1991a). The complexity of concept languages. In Allen, J., Fikes, R., & Sandewall, E. (Eds.), *Proc. of KR-91*, pp. 151–162. Morgan Kaufmann, Los Altos.

Donini, F. M., Lenzerini, M., Nardi, D., & Nutt, W. (1991b). Tractable concept languages. In *Proc. of IJCAI-91*, pp. 458–463 Sydney.

Donini, F. M., Lenzerini, M., Nardi, D., & Schaerf, A. (1994). Deduction in concept languages: From subsumption to instance checking. *J. of Logic and Computation*, *4*(4), 423–452.

Donini, F. M., Lenzerini, M., Nardi, D., & Schaerf, A. (1996). Reasoning in description logics. In Brewka, G. (Ed.), *Foundation of Knowledge Representation*. Cambrige University Press. To appear.

Doyle, J., & Patil, R. S. (1991). Two theses of knowledge representation: Language restrictions, taxonomic classification, and the utility of representation services. *AIJ*, *48*, 261–297.

Fischer, M. J., & Ladner, R. E. (1979). Propositional dynamic logic of regular programs. *J. of Computer and System Sciences*, *18*, 194–211.

Kozen, D., & Tiuryn, J. (1990). Logics of programs. In Leeuwen, J. V. (Ed.), *Handbook of Theoretical Computer Science - Formal Models and Semantics*, pp. 789–840. Elsevier.

Levy, A. Y., Rajaraman, A., & Ordille, J. J. (1996). Query answering algorithms for information agents. In *Proc. of AAAI-96*.

Nebel, B. (1988). Computational complexity of terminological reasoning in BACK. *AIJ*, *34*(3), 371–383.

Nebel, B. (1991). Terminological cycles: Semantics and computational properties. In Sowa, J. F. (Ed.), *Principles of Semantic Networks*, pp. 331–361. Morgan Kaufmann, Los Altos.

Patel-Schneider, P. F. (1989). A four-valued semantics for terminological logic. *AIJ*, *38*(1), 319–351.

Schaerf, A. (1994). Reasoning with individuals in concept languages. *Data and Knowledge Engineering*, *13*(2), 141–176.

Schild, K. (1991). A correspondence theory for terminological logics: Preliminary report. In *Proc. of IJCAI-91*, pp. 466–471 Sydney, Australia.

Schild, K. (1994). Terminological cycles and the propositional μ-calculus. In Doyle, J., Sandewall, E., & Torasso, P. (Eds.), *Proc. of KR-94*, pp. 509–520 Bonn. Morgan Kaufmann, Los Altos.

Schmidt-Schauß, M. (1989). Subsumption in KL-ONE is undecidable. In Brachman, R. J., Levesque, H. J., & Reiter, R. (Eds.), *Proc. of KR-89*, pp. 421–431. Morgan Kaufmann, Los Altos.

Schmidt-Schauß, M., & Smolka, G. (1991). Attributive concept descriptions with complements. *AIJ*, *48*(1), 1–26.

Sheth, A., Gala, S., & Navathe, S. (1993). On automatic reasoning for schema integration. *J. of Intelligent and Cooperative Information Systems*, *2*(1), 23–50.

Weida, R., & Litman, D. (1992). Terminological reasoning with constraint networks and an application to plan recognition. In *Proc. of KR-92*, pp. 282–293. Morgan Kaufmann, Los Altos.

Woods, W. A., & Schmolze, J. G. (1992). The KL-ONE family. In Lehmann, F. W. (Ed.), *Semantic Networks in Artificial Intelligence*, pp. 133–178. Pergamon Press. Published as a special issue of *Computers & Mathematics with Applications*, Volume 23, Number 2–9.

Number Restrictions on Complex Roles in Description Logics: A Preliminary Report

Franz Baader and **Ulrike Sattler***
LuFG Theoretische Informatik, RWTH Aachen, Ahornstr. 55, 52074 Aachen, Germany
email: {baader,uli}@cantor.informatik.rwth-aachen.de

Abstract

Number restrictions are concept constructors that are available in almost all implemented description logic systems. However, even though there has lately been considerable effort on integrating expressive role constructors into description logics, the roles that may occur in number restrictions are usually of a very restricted type. Until now, only languages with number restrictions on atomic roles and inversion of atomic roles, or with number restrictions on intersection of atomic roles have been investigated in detail.

In the present paper, we increase the expressive power of description languages by allowing for more complex roles in number restrictions. As role constructors, we consider composition of roles (which will be present in all our languages), and intersection, union and inversion of roles in different combinations. We will present two decidability results (for the basic language that extends $\mathcal{ALC}$ by number restrictions on roles with composition, and for one extension of this language), and three undecidability results for three other extensions of the basic language.

1 Motivation and introduction

Description logics is a field of knowledge representation in which there is a rather close interaction between theory and practice. On the one hand, there are various implemented systems based on description lcgics, which offer a palette of description formalisms with differing expressive power [Peltason,1991; Brachman *et al.*,1991; MacGregor,1991; Mays *et al.*,1991; Baader *et al.*,1994; Bresciani *et al.*,1995]. On the other hand, the computational properties (like decidability, complexity) of various description formalisms have thoroughly been investigated [Nebel,1988; Schmidt-Schauss,1989; Patel-Schneider,1989; Donini *et al.*,1991a; 1991b]. These investigation were often motivated by the use of certain constructors in systems or the need for these constructors in specific applications [Baader & Hanschke,1993; Franconi,1994], and the results have influenced the design of new systems.

The terminological formalisms of knowledge representation systems based on description logics provide constructors that can be used to build complex concepts and roles out of atomic concepts (unary predicates) and roles (binary predicates). Until recently, the main emphasis, both in implemented systems and in theoretical research, was on constructors for building complex concepts. The need for rich role constructors in certain application domains (such as representing rich schema languages for databases [Calvanese *et al.*,1994; 1995], or domains that require the appropriate modeling of part-whole relations [Padgham & Lambrix,1994; Artale *et al.*,1994; Sattler,1995]) has triggered research on description languages that also provide for expressive role constructors [Baader,1990; De Giacomo & Lenzerini,1995]. These investigations were facilitated by the observation that the formalisms considered in description logics are very similar to certain modal logics [Schild,1991; De Giacomo & Lenzerini,1994]. In particular, well-known modal logics, such as propositional dynamic logics (PDL) and its extensions [Fischer & Ladner,1979; Ben-Ari *et al.*,1982; Harel,1984], provide for role constructors like composition, union, transitive closure, and inversion.

Number restrictions are concept constructors that are available in almost all implemented description logic systems. They allow to restrict the number of role successors of an individual w.r.t. a given role. For example, if `has-child` is an atomic role and `person` is an atomic concept, then we can describe all persons having at most 2 children by the concept `person` $\sqcap$ ($\leq$ 2 `has-child`). In contrast to the rather prominent rôle that number restrictions play in de-

*This author is supported by the Deutsche Forschungsgemeinschaft under Grant No. Sp 230\ 6-6.

scription logics, the corresponding constructors in modal logic—so-called "graded modalities" [Fine,1972; van der Hoek & De Rijke,1995]—have been studied only recently, and thus there are not many results available that could be transferred to description logics. In [De Giacomo & Lenzerini,1994], the problem of adding number restrictions to PDL and various of its extensions has been investigated in detail. However, even though the description languages considered in this work have very expressive formalisms for constructing complex roles, the roles that may occur in number restrictions are restricted to atomic roles and their inverse. To the best of our knowledge, the only other well-investigated concept description language with number restrictions on non-atomic roles is $\mathcal{ALCNR}$, which allows for intersection of roles in number restrictions.

The present paper is a first attempt to overcome this research deficit. It considers description languages that extend $\mathcal{ALC}$ or $\mathcal{ALC}^+$ (the description logic equivalent to PDL) with number restrictions on complex roles. As role constructors in number restrictions, we will allow for composition (which will be present in all our languages), and intersection, union, and inversion of roles in different combinations. Number restrictions on roles with composition are particularly interesting from a practical point of view since they allow to impose restrictions on role successors for a composed role without explicitly stating restrictions on its atomic components. For example, the restriction $\texttt{person} \sqcap (= 17\ \texttt{has-child} \circ \texttt{has-child})$ describes persons that have 17 grandchildren without explicitly saying anything about the number of children, and the number of children of each child. From a theoretical point of view, number restrictions on roles with composition introduce a new level of complexity: the tree model property (which most of the modal logics and description logics investigated in the literature have) is no longer satisfied (see Section 3.1). By adding inversion of roles, we can express that a person has at least 5 siblings: $\texttt{person} \sqcap (\geq 6\ \texttt{has-child}^{-1} \circ \texttt{has-child})$; intersection of roles can prohibit that a parent marries his/her own child: $(\leq 0\ \texttt{has-child} \sqcap \texttt{is-married-to})$; union and composition can be used to describe that all children have the same name as their parent:

$$(= 1\ \texttt{has-name} \sqcup (\texttt{has-child} \circ \texttt{has-name})).$$

Number restrictions on complex roles are not only of interest in toy examples like the family domain used above. Our original motivation for considering these constructs comes from a process engineering application, where planning and optimization of large chemical plants is supported by building process models. The engineering knowledge concerning standard building blocks of these models is to be represented in a description logic system. For example, the concept $(\texttt{device} \sqcap (= 1\ \texttt{controlled-by}))$ describes devices that are controlled by a single control unit. If we want to describe a device such that all devices connected to it are controlled by the same control unit, we need composition in the number restriction: $(\texttt{device} \sqcap (= 1\ \texttt{connected-to} \circ \texttt{controlled-by}))$. To assure that the device itself is also controlled by the same unit controlling the devices connected to it, we additionally need union in the number restriction: $(\texttt{device} \sqcap (= 1\ \texttt{controlled-by} \sqcup \texttt{connected-to} \circ \texttt{controlled-by}))$. Inversion of roles comes in if we need the role `controls` as well. There are also more complex properties of devices and other parts of process models that could be expressed with number restrictions on complex roles. However, to be useful in practice, it is not sufficient to have a description language that can just be used to represent the relevant properties of objects. The description logic system must also be able to reason about the descriptions.

As a positive result in this direction, we show that the subsumption and the satisfiability problem for the language $\mathcal{ALCN}(\circ)$, which extends $\mathcal{ALC}$ with number restrictions on roles built with composition, are decidable. On the other hand, three extensions of this language turn out to be undecidable: $\mathcal{ALC}^+$ with number restrictions on roles built with composition and union; $\mathcal{ALC}$ with number restrictions on roles built with composition and intersection; and $\mathcal{ALC}$ with number restrictions on roles built with composition, union, and inversion. However, if union and intersection are restricted to role chains of the same length, then we obtain a decidable extension of $\mathcal{ALC}$.

In the next section, we introduce syntax and semantics of the concept and role constructors that will be considered. Section 3.1 describes the algorithm that decides satisfiability of $\mathcal{ALCN}(\circ)$-concepts, and Section 3.2 extends this decidability result to number restrictions on union and intersection of role chains of the same length. The subsequent section sketches the undecidability proofs, which all use a reduction of the domino problem. In Section 5, we mention related decidability and undecidability results from modal and description logics.

2 Concept and role constructors

We define syntax and semantics of all the constructors considered in the present paper, and introduce the description languages that will be investigated in more detail.

Definition 1 *Starting with* atomic roles *from a set N_R of role names,* complex roles *are built using the role constructors composition ($R \circ S$), union ($R \sqcup S$), intersection ($R \sqcap S$), inversion (R^{-1}), and transitive closure (R^+).*

The set of $\mathcal{ALC}$-concepts *is built from a set N_C of concept names using the concept constructors disjunction ($C \sqcup D$), conjunction ($C \sqcap D$), negation ($\neg C$),*

value restriction ($\forall R.C$), *and existential restriction* ($\exists R.C$), *where the roles occurring in value restrictions and existential restrictions are atomic roles. In* $\mathcal{ALC}^+_-$ concepts, *the roles occurring in value restrictions and existential restrictions may be complex roles that are built using the constructors composition, union, and transitive closure.*

Number restrictions *are concepts of the form* ($\geq n\ R$) *or* ($\leq n\ R$), *where* $n \in \mathbf{N}$ *is a nonnegative integer and* R *is a complex role. For a set* $M \subseteq \{\sqcup, \sqcap, \circ, {}^{-1}, {}^{+}\}$ *of role constructors, we call such a number restriction an* M-number restrictions *iff* R *is built using only constructors from* M. *The set of* $\mathcal{ALCN}(M)$-concepts *(resp.* $\mathcal{ALC}^+\mathcal{N}(M)$-concepts*) is obtained from* $\mathcal{ALC}$*-concepts (resp.* $\mathcal{ALC}^+_-$*-concepts) by additionally allowing for* M*-number restrictions in concepts.*

As usual in description logics, the extensions of concepts and roles involving the constructors introduced above are defined inductively on the structure of complex concepts and roles.

Definition 2 *An* interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *consists of a set* $\Delta^{\mathcal{I}}$, *called the* domain *of* $\mathcal{I}$, *and an extension function* $\cdot^{\mathcal{I}}$ *that maps every concept to a subset of* $\Delta^{\mathcal{I}}$, *and every (complex) role to a subset of* $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ *such that the followings equalities are satisfied:*

$$
\begin{aligned}
(R_1 \sqcap R_2)^{\mathcal{I}} &= {R_1}^{\mathcal{I}} \cap {R_2}^{\mathcal{I}},\\
(R_1 \sqcup R_2)^{\mathcal{I}} &= {R_1}^{\mathcal{I}} \cup {R_2}^{\mathcal{I}},\\
(R_1 \circ R_2)^{\mathcal{I}} &= \{(d,f) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \exists e \in \Delta^{\mathcal{I}} :\\
&\qquad (d,e) \in R_1^{\mathcal{I}} \wedge (e,f) \in R_2^{\mathcal{I}}\},\\
(R^{-1})^{\mathcal{I}} &= \{(e,d) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (d,e) \in R^{\mathcal{I}}\},\\
(R^{+})^{\mathcal{I}} &= \cup_{i \geq 1} (R^{\mathcal{I}})^i,\\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}},\\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}},\\
\neg C^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}},\\
(\exists R.C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \exists e \in \Delta^{\mathcal{I}} :\\
&\qquad (d,e) \in R^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\},\\
(\forall R.C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \forall e \in \Delta^{\mathcal{I}} :\\
&\qquad (d,e) \in R^{\mathcal{I}} \Rightarrow e \in C^{\mathcal{I}}\},\\
(\geq n\ R)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \#\{e \in \Delta^{\mathcal{I}} \mid (d,e) \in R^{\mathcal{I}}\} \geq n\},\\
(\leq n\ R)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \#\{e \in \Delta^{\mathcal{I}} \mid (d,e) \in R^{\mathcal{I}}\} \leq n\}.
\end{aligned}
$$

Here $\#X$ *denotes the size of a set* X. *If* $d \in C^{\mathcal{I}}$, *we say that* d *is an* instance of C *in* $\mathcal{I}$. *If* $(d,e) \in R^{\mathcal{I}}$, *we say that* d *is an* R*-predecessor of* e, *and* e *is an* R*-successor of* d *in* $\mathcal{I}$.

A concept C is called *satisfiable* iff there is some interpretation $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$. We call such an interpretation a *model of* C. A concept D *subsumes* a concept C (written $C \sqsubseteq D$) iff for all interpretations $\mathcal{I}$ we have $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Since all the languages considered in the present paper allow for negation and conjunction of concepts, subsumption and (un)satisfiability can be reduced to each other:

- $C \sqsubseteq D$ iff $C \sqcap \neg D$ is unsatisfiable,
- C is unsatisfiable iff $C \sqsubseteq A \sqcap \neg A$ (for a concept name A).

For this reason, we may restrict our attention to the satisfiability problem, both in the decidability and in the undecidability proofs.

3 Decidability results

In the first part of this section, we show that satisfiability of $\mathcal{ALCN}(\circ)$-concepts is decidable. This result is extended in the second part to a description language where, additionally, union and intersection of role chains of the same length are allowed in number restrictions.

3.1 $\mathcal{ALCN}(\circ)$ is decidable

We present a tableau-like algorithm for deciding satisfiability of $\mathcal{ALCN}(\circ)$-concepts. The algorithm and the proof of its correctness are very similar to existing algorithms and proofs for languages with number restrictions on atomic roles [Hollunder *et al.*,1990; Hollunder & Baader,1991]. It should be noted, however, that the presence of number restrictions on role chains of the form $R_1 \circ R_2 \circ \ldots \circ R_n$ with $n > 1$ has as consequence that the finite models generated by the algorithm need no longer be tree models. A *tree model* of a concept C is an interpretation such that *(1)* every element of the model can be reached from an initial (root) element, which is an instance of C, via role chains, *(2)* the root does not have a role predecessor, and *(3)* every other element has exactly one role predecessor. The following $\mathcal{ALCN}(\circ)$-concept is satisfiable, but it obviously does not have a tree model:

$$(\exists R.A) \sqcap (\exists R.\neg A) \sqcap (\forall R.(\exists S.B)) \sqcap (\leq 1\ R \circ S).$$

Nevertheless, the models that will be generated by our algorithm are very similar to tree models in that properties (1) and (2) are still satisfied, and every role chain from the root to an element has the same length (even though there may exist more than one such chain). This fact will become important in the proof of termination.

As usual, we assume without loss of generality that all concepts are in negation normal form (NNF), i.e., negation occurs only immediately in front of concept names. The basic data structure our algorithm works on are constraints:

Definition 3 *Let* $\tau = \{x, y, z, \ldots\}$ *be a countably infinite set of individual variables. A* constraint *is of the form*

$$xRy, \quad x:D, \quad \text{or} \quad x \neq y,$$ [1]

[1] We consider such inequalities as being symmetric, i.e., if $x \neq y$ belongs to a constraint system, then $y \neq x$ (implicitly) belongs to it as well.

where R is a role name, x, y are individual variables, and D is an $\mathcal{ALCN}(\circ)$-concept in NNF. A constraint system *is a set of constraints. For a constraint system S, let $\tau_S \subseteq \tau$ denote the individual variables occuring in S.*

An interpretation $\mathcal{I}$ is a model of a constraint system *S iff there is a mapping $\pi : \tau_S \to \Delta^{\mathcal{I}}$ such that $\mathcal{I}, \pi$ satisfy each constraint in S, i.e.,*

$$\begin{aligned} (\pi(x), \pi(y)) &\in R^{\mathcal{I}} && \text{for all } xRy \in S, \\ \pi(x) &\neq \pi(y) && \text{for all } (x \neq y) \in S, \\ \pi(x) &\in D^{\mathcal{I}} && \text{for all } x : D \in S. \end{aligned}$$

*For a constraint system S, individual variables x, y, and role names R_i, we say that y is an $R_1 \circ \ldots \circ R_m$-*successor *of x in S iff there are $y_0, \ldots, y_m \in \tau$ such that $x = y_0, y = y_m$, and $\{y_i R_{i+1} y_{i+1} \mid 0 \leq i \leq m - 1\} \subseteq S$. S contains a* clash *iff $\{x : A, x : \neg A\} \subseteq S$ for some concept name A and some variable $x \in \tau_S$, or $x : (\leq n\ R) \in S$ and x has $\ell > n$ R-successors $y_1, \ldots, y_\ell$ in S such that for all $i \neq j$ we have $y_i \neq y_j \in S$. A constraint system S is called* complete *iff none of the completion rules given in Figure 1 can be applied to S. In these rules, the constraint system $S[y_2/y_1]$ is obtained from S by substituting each occurrence of y_2 in S by y_1.*

Figure 1 introduces the *completion rules* that are used to test $\mathcal{ALCN}(\circ)$-concepts for satisfiability. The *completion algorithm* works on a tree where each node is labelled with a constraint system. It starts with the tree consisting of a root labelled with $S = \{x_0 : C_0\}$, where C_0 is the $\mathcal{ALCN}(\circ)$-concept in NNF to be tested for satisfiability. A rule can only be applied to a leaf labelled with a clash-free constraint system. Applying a rule $S \to S_i$, for $1 \leq i \leq n$, to such a leaf leads to the creation of n new successors of this node, each labelled with one of the constraint systems S_i. The algorithm terminates if none of the rules can be applied to any of the leaves. In this situation, it answers with "C_0 is satisfiable" iff one of the leaves is labelled with a clash-free constraint system.

Correctness of this algorithm is an immediate consequence of the following facts:

Lemma 4 *Let C_0 be an $\mathcal{ALCN}(\circ)$-concept in NNF, and let S be a constraint system obtained by applying the completion rules to $\{x_0 : C_0\}$. Then*

1. *For each completion rule $\mathcal{R}$ that can be applied to S, and for each interpretation $\mathcal{I}$ we have $\mathcal{I}$ is a model of S iff $\mathcal{I}$ is a model of one of the systems S_i obtained by applying $\mathcal{R}$.*
2. *If S is a complete and clash-free constraint system, then S has a model.*
3. *If S contains a clash, then S does not have a model.*
4. *The completion algorithm terminates when applied to $\{x_0 : C_0\}$.*

Indeed, termination shows that after finitely many steps we obtain a tree such that all its leaf nodes are labelled with complete constraint systems. If C_0 is satisfiable, then $\{x_0 : C_0\}$ is also satisfiable, and thus one of the complete constraint systems is satisfiable by (1). By (3), this system must be clash-free. Conversely, if one of the complete constraint systems is clash-free, then it is satisfiable by (2), and because of (1) this implies that $\{x_0 : C_0\}$ is satisfiable. Consequently, the algorithm is a decision procedure for satisfiability of $\mathcal{ALCN}(\circ)$-concepts:

Theorem 5 *Subsumption and satisfiability of $\mathcal{ALCN}(\circ)$-concepts is decidable.*

Proof of Part 1 of Lemma 4: We consider only the rules concerned with number restrictions, since the proof for Rules 1–4 is just as for $\mathcal{ALC}$.

5. **Number restriction:** Assume that the rule is applied to the constraint $x : (\geq n\ R_1 \circ \ldots \circ R_m)$, and that its application yields

$$\begin{aligned} S' = S \cup \{&xR_1y_2, y_mR_mz\} \\ &\cup \{y_iR_iy_{i+1} \mid 2 \leq i \leq m-1\} \\ &\cup \{z \neq w \mid w \text{ is an} \\ &\qquad R_1 \circ \ldots \circ R_m\text{-successor of } x \text{ in } S\}. \end{aligned}$$

   Since S is a subset of S', any model of S' is also a model of S.
   Conversely, assume that $\mathcal{I}$ is a model of S, and let $\pi : \tau_S \to \Delta^{\mathcal{I}}$ be the corresponding mapping of individual variables to elements of $\Delta^{\mathcal{I}}$. On the one hand, since $\mathcal{I}$ satisfies $x : (\geq n\ R_1 \circ \ldots \circ R_m)$, $\pi(x)$ has at least n $R_1 \circ \ldots \circ R_m$-successors in $\mathcal{I}$. On the other hand, since Rule 5 is applicable to $x : (\geq n\ R_1 \circ \ldots \circ R_m)$, x has less than n $R_1 \circ \ldots \circ R_m$-successors in S. Thus, there exists an $R_1 \circ \ldots \circ R_m$-successor b of $\pi(x)$ in $\mathcal{I}$ such that $b \neq \pi(w)$ for all $R_1 \circ \ldots \circ R_m$-successors w of x in S. Let $b_2, \ldots, b_m \in \Delta^{\mathcal{I}}$ be such that $(\pi(x), b_2) \in R_1^{\mathcal{I}}, (b_2, b_3) \in R_2^{\mathcal{I}}, \ldots, (b_m, b) \in R_m^{\mathcal{I}}$. We define $\pi' : \tau_{S'} \to \Delta^{\mathcal{I}}$ by $\pi'(y) := \pi(y)$ for all $y \in \tau_S$, $\pi'(y_i) := b_i$ for all $i, 2 \leq i \leq m$, and $\pi'(z) := b$. Obviously, $\mathcal{I}, \pi'$ satisfy S'.

6. **Number restriction:** Assume that the rule can be applied to $x : (\leq n\ R_1 \circ \ldots \circ R_m) \in S$, and let $\mathcal{I}$ together with the valuation $\pi : \tau_S \to \Delta^{\mathcal{I}}$ be a model of S. On the one hand, since the rule is applicable, x has more than n $R_1 \circ \ldots \circ R_m$-successors in S. On the other hand, $\mathcal{I}, \pi$ satisfy $x : (\leq m\ R_1 \circ \ldots \circ R_m) \in S$, and thus there are two different $R_1 \circ \ldots \circ R_m$-successors y_1, y_2 of x in S such that $\pi(y_1) = \pi(y_2)$. Obviously, this implies that $(y_1 \neq y_2) \notin S$, which shows that $S_{y_1,y_2} = S[y_2/y_1]$ is one of the constraint systems obtained by applying Rule 6 to $x : (\leq n\ R_1 \circ \ldots \circ R_m)$. In addition,

1. Conjunction: If $x:(C_1 \sqcap C_2) \in S$ and $x:C_1 \notin S$ or $x:C_2 \notin S$, then
$S \to S \cup \{x:C_1, x:C_2\}$

2. Disjunction: If $x:(C_1 \sqcup C_2) \in S$ and $x:C_1 \notin S$ and $x:C_2 \notin S$, then
$S \to S_1 = S \cup \{x:C_1\}$
$S \to S_2 = S \cup \{x:C_2\}$

3. Value restriction: If $x:(\forall R.C) \in S$ for a role name R, y is an R-successor of x in S and $y:C \notin S$, then
$S \to S \cup \{y:C\}$

4. Existential restriction: If $x:(\exists R.C) \in S$ for a role name R and there is no R-successor y of x in S with $y:C \in S$, then
$S \to S \cup \{xRz, z:C\}$ for a new variable $z \in \tau \setminus \tau_S$.

5. Number restriction: If $x:(\geq n\ R_1 \circ \ldots \circ R_m) \in S$ for role names $R_1, \ldots, R_m$ and x has less than n $R_1 \circ \ldots \circ R_m$-successors in S, then
$S \to S \cup \{xR_1y_2, y_mR_mz\} \cup \{y_iR_iy_{i+1} \mid 2 \leq i \leq m-1\} \cup$
$\{z \neq w \mid w$ is an $R_1 \circ \ldots \circ R_m$-successor of x in $S\}$
where z, y_i are new variables in $\tau \setminus \tau_S$.

6. Number restriction: If $x:(\leq n\ R_1 \circ \ldots \circ R_m) \in S$, x has more than n $R_1 \circ \ldots \circ R_m$-successors in S, and there are $R_1 \circ \ldots \circ R_m$-successors y_1, y_2 of x in S with $(y_1 \neq y_2) \notin S$, then
$S \to S_{y_1,y_2} = S[y_2/y_1]$
for all pairs y_1, y_2 of $R_1 \circ \ldots \circ R_m$-successors of x with $(y_1 \neq y_2) \notin S$.

Figure 1: The completion rules for $\mathcal{ALCN}(\circ)$

since $\pi(y_1) = \pi(y_2)$, $\mathcal{I}, \pi$ satisfy S_{y_1,y_2}.
Conversely, assume that $S_{y_1,y_2} = S[y_2/y_1]$ is obtained from S by applying Rule 6, and let $\mathcal{I}$ together with the valuation π be a model of S_{y_1,y_2}. If we take a valuation π' that coincides with π on the variables in $\tau_{S_{y_1,y_2}}$ and satisfies $\pi'(y_2) = \pi(y_1)$, then $\mathcal{I}, \pi'$ obviously satisfy S. ∎

Proof of Part 2 of Lemma 4: Let S be a complete and clash-free constraint system that is obtained by applying the completion rules to $\{x_0:C_0\}$. We define a canonical model $\mathcal{I}$ of S as follows:

$$\begin{array}{lll} \Delta^{\mathcal{I}} := \tau_S & \text{and} & \\ \text{for all } A \in N_C: & x \in A^{\mathcal{I}} & \text{iff } x:A \in S, \\ \text{for all } R \in N_R: & (x,y) \in R^{\mathcal{I}} & \text{iff } xRy \in S. \end{array}$$

In addition, let $\pi : \tau_S \to \Delta^{\mathcal{I}}$ be the identity on τ_S. We show that $\mathcal{I}, \pi$ satisfy every constraint in S.

By definition of $\mathcal{I}$, a role constraint of the form xRy is satisfied by $\mathcal{I}, \pi$ iff $xRy \in S$. More generally, y is an $R_1 \circ \ldots \circ R_m$-successor of x in S iff y is an $R_1 \circ \ldots \circ R_m$-successor of x in $\mathcal{I}$. We show by induction on the structure of the concept C that every concept constraint $x:C \in S$ is satisfied by $\mathcal{I}, \pi$. Again, we restrict our attention to number restrictions since the induction base and the treatment of the other constructors is just as for $\mathcal{ALC}$.

- Consider $x:(\geq n\ R_1 \circ \ldots \circ R_m) \in S$. Since S is complete, Rule 5 cannot be applied to $x:(\geq n\ R_1 \circ \ldots \circ R_m)$, and thus x has at least n $R_1 \circ \ldots \circ R_m$-successors in S, which are also $R_1 \circ \ldots \circ R_m$-successors of x in $\mathcal{I}$. This shows that $\mathcal{I}, \pi$ satisfy $x:(\geq n\ R_1 \circ \ldots \circ R_m)$.
- Constraints of the form $x:(\leq n\ R_1 \circ \ldots \circ R_m) \in S$ are satisfied because S is clash-free and complete. In fact, assume that x has more than n $R_1 \circ \ldots \circ R_m$-successors in $\mathcal{I}$. Then x also has more than n $R_1 \circ \ldots \circ R_m$-successors in S. If S contained inequality constraints $y_i \neq y_j$ for all these successors, then we would have a clash. Otherwise, Rule 6 could be applied. ∎

Proof of Part 3 of Lemma 4: Assume that S contains a clash. If $\{x:A, x:\neg A\} \subseteq S$, then it is clear that no interpretation can satisfy both constraints. Thus assume that $x:(\leq n\ R) \in S$ and x has $\ell > n$ R-successors $y_1, \ldots, y_\ell$ in S with $(y_i \neq y_j) \in S$ for all $i \neq j$. Obviously, this implies that in any model $\mathcal{I}, \pi$ of S, $\pi(x)$ has $\ell > n$ distinct R-successors $\pi(y_1), \ldots, \pi(y_\ell)$ in $\mathcal{I}$, which shows that $\mathcal{I}, \pi$ cannot satisfy $x:(\leq n\ R)$. ∎

Proof of Part 4 of Lemma 4: In the following, we consider only constraint systems S that are obtained by applying the completion rules to $\{x_0:C_0\}$. For

a concept C, we define its and/or-size $|C|_{\sqcap,\sqcup}$ as the number of occurrences of conjunction and disjunction constructors in C. The maximal role depth $depth(C)$ of C is defined as follows:

$depth(A) := depth(\neg A) := 0$ for $A \in N_C$,

$depth(C_1 \sqcap C_2) := max\{depth(C_1), depth(C_2)\}$,

$depth(C_1 \sqcup C_2) := max\{depth(C_1), depth(C_2)\}$,

$depth(\forall R_1.C_1) := depth(\exists R_1.C_1) := 1 + depth(C_1)$,

$depth(\geq n\ R_1 \circ \ldots \circ R_m) := m$,

$depth(\leq n\ R_1 \circ \ldots \circ R_m) := m$.

For the termination proof, the following observations, which are an easy consequence of the definition of the completion rules, are important:

Lemma 6 *1. Every variable $x \neq x_0$ that occurs in S is an $R_1 \circ \ldots \circ R_m$-successor of x_0 for some role chain of length $m \geq 1$. In addition, every other role chain that connects x_0 with x has the same length.*

2. If x can be reached in S by a role chain of length m from x_0, then for each constraint $x:C$ in S, the maximal role depth of C is bounded by the maximal role depth of C_0 minus m. Consequently, m is bounded by the maximal role depth of C_0.

Let m_0 be the maximal role depth of C_0. Because of the first fact, every individual x in a constraint system S (reached from $\{x_0 : C_0\}$ by applying completion rules) has a unique role level $level(x)$, which is its distance from the root node x_0, i.e., the unique length of the role chains that connect x_0 with x. Because of the second fact, the level of each individual is an integer between 0 and m_0.

In the following, we define a mapping κ of constraint systems S to $5(m_0+1)$-tuples of nonnegative integers such that $S \to S'$ implies $\kappa(S) \succ \kappa(S')$, where $\succ$ denotes the lexicographic ordering on $5m_0$-tuples. Since the lexicographic ordering is well-founded, this implies termination of our algorithm. In fact, if the algorithm did not terminate, then there would exist an infinite sequence $S_0 \to S_1 \to \ldots$, and this would yield an infinite descending $\succ$-chain of tuples.

Thus, let S be a constraint system that can be reached from $\{x_0 : C_0\}$ by applying completion rules. We define

$$\kappa(S) := (\kappa_0, \kappa_1, \ldots, \kappa_{m_0-1}, \kappa_{m_0}),$$

where $\kappa_\ell := (k_{\ell,1}, k_{\ell,2}, k_{\ell,3}, k_{\ell,4}, k_{\ell,5})$ and the components $k_{\ell,i}$ are obtained as follows:

- $k_{\ell,1}$ is the number of individual variables x in S with $level(x) = \ell$.
- $k_{\ell,2}$ is the sum of the and/or-sizes $|C|_{\sqcap,\sqcup}$ of all constraints $x:C \in S$ such that $level(x) = \ell$ and the conjunction or disjunction rule is applicable to $x:C$ in S.
- For a constraint $x:(\geq n\ R_1 \circ \ldots \circ R_m)$, let k be the maximal cardinality of all sets M of $R_1 \circ \ldots \circ R_m$-successors of x for which $y_i \neq y_j \in S$ for all pairs of distinct elements y_i, y_j of M. We associate with $x:(\geq n\ R_1 \circ \ldots \circ R_m)$ the number $r := n \dot{-} k$, if $n \geq k$, and $r := 0$ otherwise. $k_{\ell,3}$ sums up all the numbers r associated with constraints of the form $x:(\geq n\ R_1 \circ \ldots \circ R_m)$ for variables x with $level(x) = \ell$.
- $k_{\ell,4}$ is the number of all constraints $x:(\exists R.C) \in S$ such that $level(x) = \ell$ and the existential restriction rule is applicable to $x:(\exists R.C)$ in S.
- $k_{\ell,5}$ is the number of all pairs of constraints $x:(\forall R.C)$, $xRy \in S$ such that $level(x) = \ell$ and the value restriction rule is applicable to $x:(\forall R.C)$, xRy in S.

In the following, we show for each of the rules of Figure 1 that $S \to S'$ implies $\kappa(S) \succ \kappa(S')$.

1. **Conjunction:** Assume that the rule is applied to the constraint $x:C_1 \sqcap C_2$, and let S' be the system obtained from S by its application. Let $\ell := level(x)$.
   First, we compare κ_ℓ and κ'_ℓ, the tuples respectively associated with level ℓ in S and S'. Obviously, the *first components* of κ_ℓ and κ'_ℓ agree since the number of individuals and their levels are not changed. The *second component* of κ'_ℓ is *smaller* than the second component of κ_ℓ: $|C_1 \sqcap C_2|_{\sqcap,\sqcup}$ is removed from the sum, and replaced by a number that is not larger than $|C_1|_{\sqcap,\sqcup} + |C_2|_{\sqcap,\sqcup}$ (depending on whether the top constructor of C_1 and C_2 is disjunction or conjunction, or some other constructor). Since tuples are compared with the lexicographic ordering, a decrease in this component makes sure that it is irrelevant what happens in later components.
   For the same reason, we need not consider tuples κ_m for $m > \ell$. Thus, assume that $m < \ell$. In such a tuple, the first three components are not changed by application of the rule, whereas the remaining two components remain unchanged or decrease. Such a decrease can happen if $level(y) = m$ and S contains constraints yRx, $y:(\forall R.C_i)$ (or $y:(\exists R.C_i)$).
2. **Disjunction:** This rule can be treated like the conjunction rule.
3. **Value restriction:** Assume that the rule is applied to the constraints $x:(\forall R.C), xRy$, and let S' be the system obtained from S by its application. Let $\ell := level(x)$. Obviously, this implies that $level(y) = level(x) + 1 > \ell$.
   On level ℓ, the first three components of κ_ℓ remain unchanged; the fourth remains the same, or decreases (if S contains constraints zSy and $z:(\exists S.C)$ for an individual z with $level(z) = \ell$); and the fifth decreases by at least one since the

constraints $x:(\forall R.C), xRy$ are no longer counted. It may decrease by more than one if S contains constraints zSy and $z:(\forall S.C)$ for an individual z with $level(z) = \ell$.
Because of this decrease at level ℓ, the tuples at larger levels (in particular, the one for level $level(x) + 1$, where there might be an increase), need not be considered.
The tuples of levels smaller than ℓ are not changed by application of the rule. In particular, the third component of such a tuple does not change since no role constraints or inequality constraints are added or removed.

4. **Existential restriction:** Assume that the rule is applied to the constraint $x:(\exists R.C)$, and let $S' = S \cup \{xRy, y:C\}$ be the system obtained from S by its application. Let $\ell := level(x)$. Obviously, this implies that $level(y) = level(x) + 1 > \ell$.
The first two components of κ_ℓ obviously remain unchanged. The third component may decrease (if y is the first successor for an at-least restriction) or it stays the same. Since the fourth component decreases, the possible increase of the fifth component is irrelevant.
For the same reason, the increase of the first component of $\kappa_{\ell+1}$ is irrelevant.
Tuples of levels smaller than ℓ are not increased by application of the rule. All components of such a tuple remain unchanged, with the possible exception of the third component, which may decrease.

5. **Number restriction:** Assume that the rule is applied to the constraint $x:(\geq n\ R_1 \circ \ldots \circ R_m) \in S$, let S' be the system obtained by rule application, and let $\ell = level(x)$.
As for Rule 4, the first two components of κ_ℓ remain the same. In addition, there is a decrease in the third component of κ_ℓ, since the new individual z can now be added to the maximal sets of explicitly distinct $R_1 \circ \ldots \circ R_m$-successors of x. Note that these sets were previously smaller than n (because even the set of all $R_1 \circ \ldots \circ R_m$-successors of x was smaller than n).
For this reason, the possible increase in the fifth component of κ_ℓ and in the first components of tuples of levels larger than ℓ are irrelevant.Tuples of levels smaller than ℓ are either unchanged by application of the rule, or their third component decreases.

6. **Number restriction:** Assume that the rule is applied to the constraint $x:(\leq n\ R_1 \circ \ldots \circ R_m) \in S$, let $S' = S_{y_1,y_2}$ be the system obtained by rule application, and let $\ell = level(x)$.
On level $\ell + m$, the first component of the tuple $\kappa_{\ell+m}$ decreases. Thus, possible increases in the other components of this tuple are irrelevant.
Tuples associated with smaller levels remain unchanged or decrease. In fact, since y_1 in S' has all its old constraints and the constraints of y_2 in S, some value restrictions or existential restrictions for individuals of the level immediately above level $\ell + m$ may become satisfied (in the sense that the corresponding rule no longer applies). Since no constraints are removed, previously satisfied value restrictions or existential restrictions remain satisfied. The third component of tuples of smaller level cannot increase since the individuals y_1, y_2 that have been identified were not related by inequality constraints. ∎

For languages where number restrictions may contain—in addition to composition—union or intersection of roles, an important property used in the above termination proof is no longer satisfied: It is not possible to associate each individual generated by a tableau-like procedure with a unique role level, which is its distance to the "root" individual x_0 (i.e., the instance x_0 of C_0 to be generated by the tableau algorithm). Indeed, in the concept

$$C_0 := (\exists R.\exists R.A) \sqcap (\leq 1\ R \sqcup R \circ R),$$

the number restriction enforces that an R-successor of an instance of C_0 is also an $R \circ R$-successor of this instance. For this reason, an R-successor of the root individual must be both on level 1 and on level 2, and thus the relatively simple termination argument that was used above is not available for these larger languages. However, as we shall show in Section 3.2, this termination argument can still be used if union and intersection are restricted to role chains of the same length. Without this restriction, satisfiability may become undecidable: in Section 4, we show that satisfiability is in fact undecidable for $\mathcal{ALCN}(\circ, \sqcap)$. For $\mathcal{ALCN}(\circ, \sqcup)$, decidability of satisfiability is still an open problem.

3.2 An extension of the decidability result

The algorithm given in Section 3.1 will be extended such that it can also treat union and intersection of role chains that have the same length. The proof of soundness, completeness and termination of this extended algorithm is very similar to the one for the basic algorithm, and will thus only be sketched.

In the remainder of this section, a *complex role* is

- a role chain $\mathcal{R} = R_1 \circ \ldots \circ R_n$, or
- an intersection $\mathcal{R} = R_1 \circ \ldots \circ R_n \sqcap S_1 \circ \ldots \circ S_n$ of two role chains of the same length, or
- a union $\mathcal{R} = R_1 \circ \ldots \circ R_n \sqcup S_1 \circ \ldots \circ S_n$ of two role chains of the same length.

The satisfiability algorithm is extended by adding two new rules $5a$ and $5b$ to handle number restrictions $(\geq n\ \mathcal{R})$ for complex roles with union or intersection, and by substituting rule 6 by a new rule $6'$ that is able to handle the new types of complex roles. To formulate the new rules, we must extend the notion of

5a. Number restriction: If $x{:}(\geq n\ R_1 \circ \ldots \circ R_m \sqcup S_1 \circ \ldots \circ S_m) \in S$ and x has less than n $(R_1 \circ \ldots \circ R_m \sqcup S_1 \circ \ldots \circ S_m)$-successors in S, then

$S \rightarrow S_1 = S \cup \{xR_1y_2, y_mR_mz\} \cup \{y_iR_iy_{i+1} \mid 2 \leq i \leq m-1\} \cup$
$\{z \neq w \mid w \text{ is an } (R_1 \circ \ldots \circ R_m \sqcup S_1 \circ \ldots \circ S_m)\text{-successor of } x \text{ in } S\}$

$S \rightarrow S_2 = S \cup \{xS_1y_2, y_mS_mz\} \cup \{y_iS_iy_{i+1} \mid 2 \leq i \leq m-1\} \cup$
$\{z \neq w \mid w \text{ is an } (R_1 \circ \ldots \circ R_m \sqcup S_1 \circ \ldots \circ S_m)\text{-successor of } x \text{ in } S\}$

where z, y_i are new variables in $\tau \setminus \tau_S$.

5b. Number restriction: If $x{:}(\geq n\ R_1 \circ \ldots \circ R_m \sqcap S_1 \circ \ldots \circ S_m) \in S$ and x has less than n $(R_1 \circ \ldots \circ R_m \sqcap S_1 \circ \ldots \circ S_m)$-successors in S, then

$S \rightarrow S \cup \{xR_1y_2, xS_1y_2', y_mR_mz, y_m'S_mz\} \cup \{y_iR_iy_{i+1}, y_i'S_iy_{i+1}' \mid 2 \leq i \leq m-1\} \cup$
$\{z \neq w \mid w \text{ is an } (R_1 \circ \ldots \circ R_m \sqcap S_1 \circ \ldots \circ S_m)\text{-successor of } x \text{ in } S\}$

where z, y_i', y_i are new variables in $\tau \setminus \tau_S$.

6'. Number restriction: If $x{:}(\leq n\ \mathcal{R}) \in S$ for some complex role $\mathcal{R}$, x has more than n $\mathcal{R}$-successors in S, and there are $\mathcal{R}$-successors y_1, y_2 of x in S with $(y_1 \neq y_2) \notin S$, then

$S \rightarrow S_{y_1,y_2} = S[y_2/y_1]$

for all pairs y_1, y_2 of $\mathcal{R}$-successors of x with $(y_1 \neq y_2) \notin S$.

Figure 2: The additional completion rules.

a role successor in a constraint system appropriately. Building up on the notion of a role successor for a role chain, we define:

- y is an $(R_1 \circ \ldots \circ R_n \sqcup S_1 \circ \ldots \circ S_n)$-successor of x in S iff y is an $R_1 \circ \ldots \circ R_n$-successor or an $S_1 \circ \ldots \circ S_n$-successor of x in S, and
- y is an $(R_1 \circ \ldots \circ R_n \sqcap S_1 \circ \ldots \circ S_n)$-successor of x in S iff y is an $R_1 \circ \ldots \circ R_n$-successor and an $S_1 \circ \ldots \circ S_n$-successor of x in S.

Obviously, this definition is such that role successors in S are also role successors in every model of S: if $\mathcal{I}, \pi$ satisfy S, and y is an $\mathcal{R}$-successor of x in S for a complex role $\mathcal{R}$, then $\pi(y)$ is an $\mathcal{R}$-successor of $\pi(x)$ in $\mathcal{I}$.

The new rules are described in Figure 2. The rules 5*a*, 5*b* are added to the completion rules, whereas rule 6' substitutes rule 6 in Figure 1.

To show that the new algorithm obtained this way decides satisfiability of concepts for the extended language, we must proof that the four parts of Lemma 4 still hold.

1. *Local correctness* of the rules 5*a*, 5*b* and 6' can be shown as in the proof of Part 1 of Lemma 4 above.
2. The *canonical model* induced by a complete and clash-free constraint system is defined as in the proof of Part 2 of Lemma 4. The proof that this canonical model really satisfies the constraint system is also similar to the one given there. Note that our notion of an $\mathcal{R}$-successor of a complex role $\mathcal{R}$ in a constraint system was defined such that it coincides with the notion of an $\mathcal{R}$-successor in the canonical model $\mathcal{I}$ induced by the constraint system.
3. The proof that a constraint system containing a clash is unsatisfiable is the same as the one given above. Note that this depends on the fact that role successors in a constraint system are also role successors in every model of the constraint system.
4. The proof of *termination* is also very similar to the one given above. The definition of the depth of a concept is extended in the obvious way to concepts with number restrictions on complex roles:

$$\begin{aligned} \mathit{depth}(\geq n\ R_1 \circ \ldots \circ R_m \sqcap S_1 \circ \ldots \circ S_m) &:= m, \\ \mathit{depth}(\geq n\ R_1 \circ \ldots \circ R_m \sqcup S_1 \circ \ldots \circ S_m) &:= m, \\ \mathit{depth}(\leq n\ R_1 \circ \ldots \circ R_m \sqcap S_1 \circ \ldots \circ S_m) &:= m, \\ \mathit{depth}(\leq n\ R_1 \circ \ldots \circ R_m \sqcup S_1 \circ \ldots \circ S_m) &:= m. \end{aligned}$$

Because the role chains in complex roles are of the same length, it is easy to see that Lemma 6 still holds. Thus, we can define the same measure $\kappa(S)$ as above for all constraint systems obtained by applying the extended completion rules to $\{x_0 : C_0\}$. It is easy to see that the proof that $S \rightarrow S'$ implies $\kappa(S) \succ \kappa(S')$ can be extended to the new rules. It should be noted that the proof given above was already formulated in a more general way than necessary for the language considered there. In fact, we have only used that all role chains connecting two individuals have the same length (which is still satisfied for the extended language), and not

that these role chains also have the same name (which is only satisfied for $\mathcal{ALCN}(\circ)$).

The following theorem is an immediate consequence of these observations:

Theorem 7 *Subsumption and satisfiability is decidable for the language that extends $\mathcal{ALCN}(\circ)$ by number restrictions on union and intersection of role chains of the same length.*

4 Undecidability results

We will use a reduction of the domino problem—a well-known undecidable problem [Knuth,1968; Berger,1966] often used in undecidability proofs in logic—to show that concept satisfiability is undecidable for three extensions of the decidable language $\mathcal{ALCN}(\circ)$ considered in the previous section.

Definition 8 *A* tiling system $\mathcal{D} = (D, H, V)$ *is given by a non-empty set* $D = \{D_1, \ldots, D_\ell\}$ *of* domino types, *and by horizontal and vertical* matching pairs $H \subseteq D \times D$, $V \subseteq D \times D$. *The* domino problem *asks for a* compatible tiling *of the first quadrant* $\mathbf{N} \times \mathbf{N}$ *of the plane, i.e., a mapping* $t : \mathbf{N} \times \mathbf{N} \to D$ *such that for all* $m, n \in \mathbf{N}$:

$$(t(m,n), t(m+1,n)) \in H \wedge (t(m,n), t(m,n+1)) \in V.$$

In order to reduce the domino problem to satisfiability of concepts, we must show how a given tiling system $\mathcal{D}$ can be translated into a concept $E_\mathcal{D}$ (of the language under consideration) such that $E_\mathcal{D}$ is satisfiable iff $\mathcal{D}$ allows for a compatible tiling. This task can be split into three subtasks, which we will first explain on an intuitive level, before showing how they can be achieved for the three concept languages under consideration.

Task 1: It must be possible to represent a single "square" of $\mathbf{N} \times \mathbf{N}$, which consists of points $(n,m), (n,m+1), (n+1,m)$, and $(n+1,m+1)$. The idea is to introduce roles X, Y, where X goes one step into the horizontal (i.e. x-) direction, and Y goes one step into the vertical (i.e. y-) direction. The concept language must be expressive enough to describe that an individual (a point (n,m)) has exactly one X-successor (the point $(n+1,m)$), exactly one Y-successor (the point $(n,m+1)$), and that the $X \circ Y$-successor coincides with the $Y \circ X$-successor (the point $(n+1,m+1)$).

Task 2: It must be possible to express that a tiling is locally correct, i.e., that the X- and Y-successors of a point have an admissible domino type. The idea is to associate each domino type D_i with an atomic concept D_i, and to express the horizontal and vertical matching conditions via value restrictions on the roles X, Y.

Task 3: It must be possible to impose the above *local* conditions on all points in $\mathbf{N} \times \mathbf{N}$. This can be achieved by constructing a "universal" role U and a "start" individual such that every point is a U-successor of this start individual. The local conditions can then be imposed on all points via value restrictions on U for the start individual.

Task 2 is rather easy, and can be realized using the $\mathcal{ALC}$-concept $C_\mathcal{D}$ given in Figure 3. The first conjunct expresses that every point has exactly one domino type, and the value restrictions in the second conjunct express the horizontal and vertical matching conditions.

Task 1 can be achieved in any extension of $\mathcal{ALCN}(\circ)$ with either union or intersection of roles in number restrictions: see the concepts $C_\sqcup$ and $C_\sqcap$ in Figure 3.

Task 3 is easy for languages that extend $\mathcal{ALC}^+$, and more difficult for languages without transitive closure. The general idea is that the start individual s is an instance of the concept $E_\mathcal{D}$ to be constructed. From this individual, one can reach via U the origin $(0,0)$ of $\mathbf{N} \times \mathbf{N}$, and all points that are connected with the origin via arbitrary X- and Y-chains.

(1) In extensions of $\mathcal{ALC}^+$, we can use an atomic role R to reach the origin, and the complex role $R \sqcup (R \circ (X \sqcup Y)^+)$ to reach every point. Thus, the tiling system $\mathcal{D}$ can be translated into the $\mathcal{ALC}^+\mathcal{N}(\circ, \sqcup)$-concept

$$E_\mathcal{D}^{(1)} := (= 1\ R) \sqcap (\forall (R \sqcup (R \circ (X \sqcup Y)^+)).(C_\sqcup \sqcap C_\mathcal{D})).$$

We can even restrict the complex role in the value restriction to a simple transitive closure of an atomic role. To achieve this, we make sure that the X- and the Y-successors of a point are also R-successors of this point. This allows us to use R^+ in place of $R \sqcup (R \circ (X \sqcup Y)^+)$ as "universal" role: see the concept $E_\mathcal{D}^{(1')}$ in Figure 3.

(2) In $\mathcal{ALCN}(\circ, \sqcup, ^{-1})$, we explicitly introduce a role name U for the "universal" role, and use number restrictions involving composition, union, and inversion of roles to make sure that the start individual is directly connected via U with every point: see the concept $E_\mathcal{D}^{(2)}$ in Figure 3. The number restrictions inside the value restriction make sure that every point p that is reached via U from the start individual satisfies the following: Its X-successor and its Y-successor each have exactly one U-predecessor, which coincides with the (unique) U-predecessor of p, i.e., the start individual. Thus, the X-successor and the Y-successor of p are also U-successors of the start individual.

(3) For $\mathcal{ALCN}(\circ, \sqcap)$, a similar construction is possible if we introduce role names R and T. The intuition is that T plays the rôle of the inverse of R (except for one individual), and the "universal" role corresponds to the composition $R \circ T \circ R$: The start individual s (which

$$C_{\mathcal{D}} := \bigsqcup_{1\le i\le m}(D_i \sqcap (\bigsqcap_{\substack{1\le j\le m\\ i\neq j}} \neg D_j)) \sqcap \bigsqcap_{1\le i\le m}(D_i \Rightarrow ((\forall X.(\bigsqcup_{(D_i,D_j)\in H} D_j)) \sqcap (\forall Y.(\bigsqcup_{(D_i,D_j)\in V} D_j)))),$$

$$C_{\sqcup} := (=1\ X) \sqcap (=1\ Y) \sqcap (=1\ X\circ Y) \sqcap (=1\ Y\circ X) \sqcap (=1\ Y\circ X \sqcup X\circ Y),$$

$$C_{\sqcap} := (=1\ X) \sqcap (=1\ Y) \sqcap (=1\ X\circ Y) \sqcap (=1\ Y\circ X) \sqcap (=1\ Y\circ X \sqcap X\circ Y),$$

$$E_{\mathcal{D}}^{(1')} := (=1\ R) \sqcap (\forall R^+.(C_{\sqcup} \sqcap C_{\mathcal{D}} \sqcap (\geq 2\ R) \sqcap (\leq 2\ R \sqcup X \sqcup Y)))$$

$$E_{\mathcal{D}}^{(2)} := (\geq 1\ U) \sqcap (\forall U.(C_{\sqcup} \sqcap C_{\mathcal{D}} \sqcap (=1\ X\circ U^{-1}) \sqcap (=1\ Y\circ U^{-1}) \sqcap (\leq 1\ U^{-1} \sqcup Y\circ U^{-1} \sqcup X\circ U^{-1}))).$$

$$E_{\mathcal{D}}^{(3)} := (=1\ R \sqcap R\circ T\circ R) \sqcap (\forall R.\forall T.\forall R.\ (C_{\sqcap} \sqcap C_{\mathcal{D}} \sqcap (\leq 1\ T) \sqcap (\forall Y.(\leq 1\ T)) \sqcap (\forall X.(\leq 1\ T)) \sqcap (=1\ T \sqcap X\circ T \sqcap Y\circ T) \sqcap (=1\ X \sqcap X\circ T\circ R) \sqcap (=1\ Y \sqcap Y\circ T\circ R)))$$

where $A \Rightarrow B$ is an abbreviation for $\neg A \sqcup B$ and $(=n\ R)$ is an abbreviation for $(\geq n\ R) \sqcap (\leq n\ R)$.

Figure 3: Concepts used in the undecidability proofs

is an instance of $E_{\mathcal{D}}^{(3)}$), has exactly one R-successor $p_{(0,0)}$, which coincides with its $R\circ T\circ R$-successor. The individual $p_{(0,0)}$ corresponds to the origin of $\mathbf{N}\times\mathbf{N}$. Let s' be the $R\circ T$-successor of s. The number restrictions of $E_{\mathcal{D}}^{(3)}$ make sure that $p_{(0,0)}$ satisfies the following: It has exactly one T-successor, namely s', which coincides with the (unique) T-successors of its X- and Y-successors. In addition, the (unique) X-successor of $p_{(0,0)}$ is also an $X\circ T\circ R$-successor of $p_{(0,0)}$, which makes sure that the X-successor of $p_{(0,0)}$ is an R-successor of s', and thus an $R\circ T\circ R$-successor of s. The same holds for the Y-successor. One can now continue the argument with the X-successor (resp. Y-successor) of $p_{(0,0)}$ in place of $p_{(0,0)}$.

With the intuition given above, it is not hard to show for all $i, 1\le i\le 3$, that a tiling system $\mathcal{D}$ has a compatible tiling iff $E_{\mathcal{D}}^{(i)}$ is satisfiable.

Theorem 9 *Satisfiability (and thus also subsumption) of concepts is undecidable for* $\mathcal{ALC}^+\mathcal{N}(\circ,\sqcup)$, $\mathcal{ALC}\mathcal{N}(\circ,\sqcup,^{-1})$, *and* $\mathcal{ALC}\mathcal{N}(\circ,\sqcap)$.

The concept $E_{\mathcal{D}}^{(1')}$ shows that the undecidability result for $\mathcal{ALC}^+\mathcal{N}(\circ,\sqcup)$-concepts also hold if only transitive closure of atomic roles is allowed.

5 Related work and open problems

Propositional dynamic logic (PDL), which corresponds to our language $\mathcal{ALC}^+$, has been shown to be decidable in [Fischer & Ladner,1979], and decidability of its extension by deterministic programs, DPDL, is shown in [Ben-Ari *et al.*,1982]. In principle, the use of deterministic programs corresponds to introducing a restricted form of number-restrictions, namely $(\leq 1\ R)$ for atomic roles R. Adding inversion (of atomic roles) to DPDL has a drastic consequence: the finite model property is lost, i.e., there are satisfiable formulae (concepts) that do not have a finite model. Nevertheless, satisfiability is still decidable [Vardi,1985] (EXPTIME complete, like all the other decision problems for PDL and its extensions mentioned until now). It should be noted, however, that in these languages inversion does not occur in the number-restrictions, since only atomic programs are asserted to be deterministic. In [De Giacomo,1995], general number restrictions and Boolean operators for roles are added to PDL with inversion, and (EXPTIME) decidability is shown by a rather ingeneous reduction to the decision problem for PDL. In this work, atomic roles and their inverse my occur in number restrictions, but not more complex roles. In addition, the complement of roles is built relative to a fixed role any, which need not be interpreted as the universal role. Thus, it does not yield the classical negation of roles. If one adds number restrictions on atomic roles and their intersections to $\mathcal{ALC}$, satisfiability for the obtained language is still decidable with a PSPACE-algorithm [Donini *et al.*,1991a]. Certain modal logics and concept description languages (e.g., $\mathcal{ALC}$) can be translated into first-order logic such that only two different variable names occur in the formulae obtained by this translation (see, e.g., [Borgida,1996]). Thus, decidability of subsumption and other inference problems for these languages follows from the known decidability result for $\mathcal{L}_2$, i.e., first-order logic with two variables and without function symbols [Mortimer,1975]. Recently, this decidability result has been extended to $\mathcal{C}_2$, i.e., predicate logic with 2 variables and counting quantifiers [Grädel *et al.*,]. As an immediate consequence, satisfiability and subsumption for $\mathcal{ALC}\mathcal{N}(\sqcup,\sqcap,\neg,^{-1})$, the extension of $\mathcal{ALC}$ by number restrictions with inversion and Boolean operators on roles, is still decidable. It should be

noted, however, that expressing composition of roles in predicate logic requires more than two variables.

In this paper, we have shown that the language $\mathcal{ALCN}(\circ)$, which adds number restrictions on roles with composition to $\mathcal{ALC}$, is still decidable. It is not clear, however, whether there exists a PSPACE algorithm for the problem. The one presented above is EXPTIME, and since different role paths need to be joined together, the trace method developed in [Schmidt-Schauß & Smolka,1991] cannot directly be applied. Almost all extensions of $\mathcal{ALCN}(\circ)$ by union, intersection, and inversion of roles were shown to be undecidable (unless restricted to role chains of the same length). Only decidability for $\mathcal{ALCN}(\circ, \sqcup)$ is still open. For $\mathcal{ALC}^+$, however, the extension by composition and union could already be shown to be undecidable.

As related undecidability results, one can mention undecidability of the extension of DPDL by intersection of roles (which does not occur in the number restrictions, however) [Harel,1984]. In [Hanschke,1992], an extension of $\mathcal{ALC}$ by so-called existential and universal agreements on role chains is shown to be undecidable. It is easy to see that existential (universal) agreements can be simulated by number restrictions involving composition and intersection (union).

References

[Artale *et al.*, 1994] A. Artale, F. Cesarini, E. Grazzini, F. Pippolini, and G. Soda. Modelling composition in a terminological language environment. In *Workshop Notes of the ECAI Workshop on Parts and Wholes: Conceptual Part-Whole Relations and Formal Mereology*, pages 93–101, Amsterdam, 1994.

[Baader *et al.*, 1994] F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H. Profitlich. An empirical analysis of optimization techniques for terminological representation systems, or: Making KRIS get a move on. *Applied Artificial Intelligence*, 4:109–132, 1994.

[Baader, 1990] F. Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. Technical Report RR-90-13, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Kaiserslautern, Germany, 1990. An abridged version appeared in *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence IJCAI-91*, pp. 446–451.

[Baader & Hanschke, 1993] F. Baader and P. Hanschke. Extensions of concept languages for a mechanical engineering application. In *Proc. of the 16th German AI-Conference, GWAI-92*, volume 671 of *Lecture Notes in Computer Science*, pages 132–143, Bonn, Germany, 1993. Springer-Verlag.

[Ben-Ari *et al.*, 1982] M. Ben-Ari, J. Halpern, and A. Pnueli. Deterministic propositional dynamic logic: finite models, complexity and completeness. *Journal of Computer and System Science*, 25:402–417, 1982.

[Berger, 1966] R. Berger. The undecidability of the dominoe problem. *Mem. Amer. Math. Soc.*, 66, 1966.

[Borgida *et al.*, 1995]
A. Borgida, M. Lenzerini, D. Nardi, and B. Nebel, editors. *Proceedings of the International Workshop on Description Logics*, Rome, 1995.

[Borgida, 1996] A. Borgida. On the relative expressive power of Description Logics and Predicate Calculus. To appear in *Artificial Intelligence*, 1996.

[Brachman *et al.*, 1991] R. J. Brachman, D. McGuinness, P. Patel-Schneider, L. Resnick, and A. Borgida. Living with CLASSIC: When and how to use a KL-ONE-like language. In J. F. Sowa, editor, *Principles of Semantic Networks*. Morgan Kaufmann, Los Altos, 1991.

[Bresciani *et al.*, 1995] P. Bresciani, E. Franconi, and S. Tessaris. Implementing and testing expressive description logics: A preliminary report. In Borgida et al. [1995].

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

[Calvanese *et al.*, 1995] D. Calvanese, G. De Giacomo, and M. Lenzerini. Structured objects: Modeling and reasoning. In *Proc. of the 4th International Conference on Deductive and Object-Oriented Databases (DOOD-95)*, volume 1013 of *Lecture Notes in Computer Science*, pages 229–246, 1995.

[De Giacomo, 1995] G. De Giacomo. *Decidability of Class-Based Knowledge Representation Formalisms*. PhD thesis, Università degli Studi di Roma "La Sapienza", 1995.

[De Giacomo & Lenzerini, 1994] G. De Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics (extended abstract). In *Proc. of the 12th Nat. Conf. on Artificial Intelligence (AAAI-94)*, 1994.

[De Giacomo & Lenzerini, 1995] G. De Giacomo and M. Lenzerini. 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)*, 1995.

[Donini *et al.*, 1991a] F. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In *Proc. of the 2nd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-91)*, Boston (USA), 1991.

[Donini *et al.*, 1991b] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. Tractable concept languages. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI-91)*, pages 458–463, Sydney, 1991.

[Fine, 1972] K. Fine. In so many possible worlds. *Notre Dame Journal of Formal Logics*, 13:516–520, 1972.

[Fischer & Ladner, 1979] M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Science*, 18:194–211, 1979.

[Franconi, 1994] E. Franconi. A treatment of plurals and plural quantifications based on a theory of collections. *Minds and Machines*, 3(4):453–474, November 1994.

[Grädel *et al.*,] E. Grädel, M. Otto, and E. Rosen. Two-variable logic with counting is decidable. Preprint, available via `http://www.informatik.rwth-aachen.de/WWW-math/papers.html`.

[Hanschke, 1992] P. Hanschke. Specifying Role Interaction in Concept Languages. In *Proc. of the 2nd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-91)*, 1992.

[Harel, 1984] D. Harel. Dynamic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic: Extensions of Classical Logic*, volume 2. D. Reidel, 1984.

[Hollunder *et al.*, 1990] B. Hollunder, W. Nutt, and M. Schmidt-Schauss. Subsumption algorithms for concept description languages. In *ECAI-90*, Pitman Publishing, London, 1990.

[Hollunder & Baader, 1991] B. Hollunder and F. Baader. Qualifying number restrictions in concept languages. In *Proc. of the 2nd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-91)*, pages 335–346, Boston (USA), 1991.

[Knuth, 1968] D. Knuth. *The Art of computer programming*, volume 1. Addison Wesley Publ. Co., Reading, Massachussetts, 1968.

[MacGregor, 1991] R. MacGregor. The evolving technology of classification-based knowledge representation systems. In J. F. Sowa, editor, *Principles of Semantic Networks*. Morgan Kaufmann, Los Altos, 1991.

[Mays *et al.*, 1991] E. Mays, R. Dionne, and R. Weida. K-Rep system overview. *SIGART Bulletin*, 2(3):93–97, 1991.

[Mortimer, 1975] M. Mortimer. On languages with two variables. *Zeitschr. f. math. Logik u. Grundlagen d. Math.*, 21:135–140, 1975.

[Nebel, 1988] B. Nebel. Computational complexity of terminological reasoning in BACK. *Artificial Intelligence*, 34(3):371–383, 1988.

[Padgham & Lambrix, 1994] L. Padgham and P. Lambrix. A framework for part-of hierarchies in terminological logics. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proc. of the 4th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-94)*, pages 485–496, 1994.

[Patel-Schneider, 1989] P. F. Patel-Schneider. Undecidability of subsumption in NIKL. *Artificial Intelligence Journal*, 39:263–272, 1989.

[Peltason, 1991] C. Peltason. The BACK System - an Overview. *SIGART Bulletin*, 1991.

[Sattler, 1995] U. Sattler. A concept language for an engineering application with part-whole relations. In Borgida et al. [1995], pages 119–123.

[Schild, 1991] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI-91)*, pages 466–471, Sydney, 1991.

[Schmidt-Schauss, 1989] M. Schmidt-Schauss. Subsumption in KL-ONE is undecidable. In *Proc. of the 1st Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-89)*, pages 421–431, Boston (USA), 1989.

[Schmidt-Schauß & Smolka, 1991] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[van der Hoek & De Rijke, 1995] W. van der Hoek and M. De Rijke. Counting objects. *Journal of Logic and Computation*, 5(3):325–345, 1995.

[Vardi, 1985] M. Vardi. The Taming of Converse: Reasoning about Two-Way Computations. In *Proc. of the 6th Workshop on Logic of Programs*, volume 193 of *Lecture Notes in Computer Science*, pages 413–424. Springer-Verlag, 1985.

Asking Queries about Frames

Alexander Borgida and Deborah L. McGuinness*
Dept. of Computer Science
Rutgers University
New Brunswick, NJ
{borgida,dlm}@cs.rutgers.edu

Abstract

Frame-based knowledge representation and reasoning systems typically provide procedural interfaces for asking about properties of individuals and concepts. We propose an alternative declarative approach that extends standard interface functionality by supporting selective viewing of components of complex objects. Instead of just returning sets of individuals, our queries match concepts and *filtered* fragments of descriptions. The query language is an extended form of the language used to describe the knowledge-base contents, thus facilitating user training.

In this paper, we describe a variety of possible semantics for answering queries in description logics. We investigate the algorithms required when answers are deduced by matching queries against a "structural normal form" of descriptions. As part of our approach, we introduce a useful refinement of the notion of structural subsumption.

1 Motivation

Our experience developing large knowledge bases using frame-based knowledge representation and reasoning systems (FR-KRSs) — CLASSIC [3] in our case — has revealed a significant lacuna: when displaying or explaining complex concepts and individuals, the system's built-in functions (such as **print**) flood the knowledge engineer with uninteresting and irrelevant information. For example, these functions do not differentiate widely-known "definitional" knowledge (e.g., **PEOPLE**'s **ages** should be of type **INTEGER**) from more specific constraints (the **age** value is restricted to be of type "**[min 14]**" - the set of integer values no less than 14). Also they can not distinguish special-purpose roles/slots (e.g., roles describing how to retrieve information from data bases or how to produce multimedia displays), which are of interest for different audiences. Moreover, knowledge representation and reasoning systems (KRSs) typically focus on context independent needs and neglect context dependent, domain specific requirements. Still, many applications demand attention to context and the domain. For example, although in general we might not care to see the **sugarContent** of an instance **f** of the **FOOD** class, this becomes crucial when **f** appears as filler for the **eats** role of a diabetic person. Currently, such difficulties can be resolved only by writing one-of-a-kind special-purpose procedures, which use the application program interface (API) to the KRS. For example, one can retrieve the value restriction on the **age** role by invoking **get-value-restriction(Joe,age)**; and then check whether the answer returned is strictly more specialized than the concept **INTEGER**, by using the function **concept-subsumes**.

This is symptomatic of a more general problem with KRSs in general, and Description Logics (DLs) in particular: Considerable effort has been expended in looking for "good" languages in which to make assertions to be stored in the knowledge base (for so-called TELL operations) — languages which, for example, permit only statements whose logical consequences are effectively computable. In contrast, very little attention has been paid to languages for asking questions (or for describing answers), leaving this aspect to the procedural interface. This reliance on a procedural solution, rather than a declarative solution, is undesirable for several reasons:

1. *Declarative* queries are preferable since they are more concise, can be analyzed (e.g., for coherence), can be optimized, etc. The advantages of powerful declarative query languages have in fact long been recognized and exploited in database systems.
2. Declarative queries can also be cached, organized and reused more easily, which is increasingly important for tasks such as data exploration and op-

*AT&T Labs-Research, Murray Hill, NJ 07974, dlm@research.att.com

timization [4, 5].

3. A query notation unrelated to either the TELL language or to the structure of the objects is harder for users to learn.

4. Even powerful query languages based on First Order Predicate Calculus (FOPC), such as in LOOM[13], do not to support questions returning concepts and formulas, such as the ones provided by the API procedures.

We will propose a declarative language for querying FR-KRSs that addresses many of the above issues. In particular, queries will resemble concepts in a natural way, and answers to queries can be concepts, or even new descriptions in DLs, not just individual values. We will show with examples that the precise notion of "desired answer" is in fact not obvious, and will choose to pursue our own research in the paradigm of concepts in "structural subsumption" normal-form. In order to develop and study query-answering algorithms, we will need to take a detour to refine the notion of structural subsumption introduced in [6]. Finally, we will describe some enhancements to the query language which have been motivated by our initial practical applications [19, 4].

An earlier design, which lead in part to the ideas in this paper, has been implemented for the CLASSIC system [18] and has been used in an application to greatly simplify the presentations and explanations of configurations [15].

2 Concepts and Queries

Frame-based knowledge base management systems represent information using individuals, grouped into concepts, and inter-related by binary relationships called slots or roles. Concepts are related by a subsumption/IsA relationship, written here as $\Longrightarrow$, while individuals are instances of concepts, written as $\rightarrow$. Thus, we will write **PERSON** $\Longrightarrow$ **MAMMAL** and **Charles** $\rightarrow$**PERSON**.

2.1 Description Logics

For concreteness, we shall use the syntax of a DL resembling CLASSIC, which supports both primitive concepts, such as **BEAR**, and composite ones. Structured, composite concepts are built with so-called concept constructors, and may be, among others, enumerations of individuals (e.g., [**one-of Amy Lulu**]), or number and type restrictions on roles. For example, the concept **PICKY-EATER**, defined as [**and** [**at-most** 2 **eats**] [**all eats SWEET-FOOD**]], is intended to denote individuals that have no more than two fillers for the **eats** role, and all such fillers must be instances of the concept **SWEET-FOOD**, defined elsewhere.

```
Concept ::=
        ConceptName |
        [one-of Ind1 ... Indn] |
        [all Role Concept] |
        [at-least Integer Role] |
        [at-most Integer Role] |
        [fills Role Ind] |
        [min Integer] |
        [max Integer] |
        [and Concept ... Concept]
ConceptName ::= Defined Concept Identifier |
        Primitive Concept Identifier
Role ::= Role Identifier
Ind ::= Individual Identifier
```

Table 1: Syntax of MiniDL

CONCEPT	DENOTATION
primitive A	$A^{\mathcal{I}}$
[one-of B_1,...,B_m]	$\{B_1^{\mathcal{I}}, \ldots, B_m^{\mathcal{I}}\}$
[min n]	$\{d \in \mathcal{N} \mid d \geq n\}$
[max n]	$\{d \in \mathcal{N} \mid d \leq n\}$
[all p C]	$\{d \in \Delta^{\mathcal{I}} \mid p^{\mathcal{I}}(d) \subseteq C^{\mathcal{I}}\}$
[some p C]	$\{d \in \Delta^{\mathcal{I}} \mid p^{\mathcal{I}}(d) \cap C^{\mathcal{I}} \neq \emptyset\}$
[at-least n p]	$\{d \in \Delta^{\mathcal{I}} \mid \lvert p^{\mathcal{I}}(d)\rvert \geq n\}$
[at-most n p]	$\{d \in \Delta^{\mathcal{I}} \mid \lvert p^{\mathcal{I}}(d)\rvert \leq n\}$
[fills p B]	$\{d \in \Delta^{\mathcal{I}} \mid B^{\mathcal{I}} \in p^{\mathcal{I}}(d)\}$
[and C D]	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$

Table 2: Semantics of MiniDL

The syntax of the language is given by the grammar in Table 1. In Table 2, we provide the denotational semantics of this DL fragment, using the notion of a valuation $\mathcal{I}$ that assigns a subset of a domain $\Delta^{\mathcal{I}}$to primitive concepts, and assigns a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to primitive roles.

Finally, a particular KB contains, in addition to concept declarations, information about the membership of individuals in concepts, and their role fillers: "Amy is a BEAR with at least 3 friends", "Lulu's age is 9". Both of these can be recorded as membership assertions: **Amy**$\rightarrow$**[and BEAR [at-least 3 friends]]**, **Lulu**$\rightarrow$**[fills age 9]**.

Systems based on DLs perform a number of operations on descriptions and individuals, such as checking for consistency and "classifying" concepts into a subsumption or IsA hierarchy, where D1 subsumes D2 iff the denotation of D1 contains that of D2 in all possible interpretations. Thus, **MAMMAL** subsumes **PERSON** if and only if every individual that is a **PERSON** is also a **MAMMAL** in all possible interpretations.

2.2 Query Concepts

The conventional knowledge-level view of asking a knowledge base KB some question is to determine whether the query formula Q is logically entailed by it: $KB \models Q$. While this yields only "Yes" or "No" as answers, a standard generalization for first-order logic KBs is to ask queries with free variables, such as $Q(?x, ?y)$; the answer is the set of valid substitutions for the variables: $\{ \sigma \mid \sigma$ maps *FREE-VARS*(Q) to constants such that $KB \models \sigma(Q) \}$, where $\sigma(Q)$ stands for Q with every variable y replaced by $\sigma(y)$. For example, given the FOPC KB { `likes(Ann,Deb)`, `likes(Ann,Honey)` }, the query `likes(Ann,?x)`, returns as answers the substitutions mapping ?x to `Deb` and `Honey` respectively (denoted by { ?x↦Deb } and { ?x↦Honey }).

This suggests that we introduce variables (here, syntactically distinguished by a preceding question mark) for frame questions as well. To achieve our goal of querying portions of frames or descriptions, we allow variables to appear anywhere identifiers or constants may appear in the syntax of concepts (viz. Table 1). For example, **[at-least ?n friends]** or **[all friends [fills liveAt ?x]]** will be query concepts. Such a query Q will be directed against a particular individual `i` (written as `i::Q`) or concept `D` (written as `D::Q`), so that the process of answering `D::Q` will resemble the notion of *matching*. When such a query concept is matched against an individual or concept, the result will be a (set of) substitution(s), or *failure*, in case no substitution can be found.

The following examples are intended to provide an initial, intuitive feeling for the notion of query answering. Consider the knowledge base KB1= {
Lulu→[and BEAR [all friends NICE]],
Lulu→[fills age 9],
Lulu→[at-least 3 friends] }.
The match **Lulu:[fills age ?x]** should naturally return the substitution ?x↦9, while the match **Lulu:[all friends ?X]** returns **?X↦NICE**.[1] More generally, one can have complex query concepts, such as

[and [at-least ?n friends]
[all friends [all age ?a]]]

which, when matched against Lulu, would result in failure, since the nested match **NICE::[all age ?a]** does not produce a substitution.

Clearly, this approach provides two of the advantages we desired: the query language mirrors the natural form of the knowledge base contents, and it allows queries that return (portions of) frames, as in **[all friend ?X]**. We do however need to be circumspect since matching is more complicated than it appears at first glance. We will proceed by first presenting a number of less obvious examples, and then examining in greater detail two approaches based on structural subsumption. We will return in the end to describe some enhancements to the query language.

In the rest of this paper we will concentrate on the notion of matching query concepts against concept descriptions, rather than individuals, both because of lack of space and because in several systems reasoning about individuals can be reduced to reasoning about concepts. (We have investigated matching query concepts against individuals in a slightly different setting in [14].)

2.3 On the semantics of matching

First, note that it would be very reasonable to expect the match **[all pets NOTHING] :: [at-most ?n pets]** to succeed with ?n↦0. The meaning of **NOTHING** is the empty set, thus **[at-most** 0 pets] is semantically equivalent to **[all pets NOTHING]**. Suppose we treat descriptions as terms, with concept constructors as functors, so that **[all pets NOTHING]** is represented as **all**(pets,**NOTHING**). A *purely syntactic* notion of matching is then clearly undesirable, since the standard unification of the terms **all**(pets,**NOTHING**) and **at-most**(?n,pets) fails.

Even if we used unification modulo a theory of concept equivalence, the result would not be satisfactory, because the match of **PICKY-EATER**, whose definition was **[and [at-most 2 eats] [all eats SWEET-FOOD]]**, against **[all eats ?X]** would fail, because the conjunct with the **at-most** restriction is missing in the pattern. This match failure would prevent us from using queries to extract interesting *parts* of concepts, as we originally wanted.

We are therefore naturally led to consider the meaning of C::Q as a request for a substitution σ such that C is *subsumed by* σ(Q). For example, given the concept **PICKY-EATER** above and the query concept **[all eats ?X]**, we can would like the substitution ?X↦**SWEET-FOOD** since this substitution applied to the query concept yields **[all eats SWEET-FOOD]**, which subsumes **PICKY-EATER**. However, even here we run into trouble because the match **PICKY-EATER::[at-most ?n eats]** would return in addition to ?n↦2, an infinite number of substitutions, ?n↦3, ?n↦4, etc., since **[at-most 3 eats]**, etc., all subsume **PICKY-EATER**. Yet, again, for the purpose of displaying part of **PICKY-EATER**, we only want to see the first substitution. The reason we would prefer the first is because of all the substitutions in the infinite set, this one, when applied to the query concept, results in the most specific concept. Therefore, we propose the following as a baseline definition:

[1] Note that variables are typed by context as matching individuals, roles, concepts, integers, or sets thereof. To facilitate reading, we use upper case variables for concepts and lower case for everything else in our examples.

Definition 1 *Given query concept Q and description D, the match D::Q succeeds with substitution σ if and only if D is subsumed by $\sigma(Q)$, and there is no σ' such that D is subsumed by $\sigma'(Q)$, with $\sigma'(Q)$ strictly subsumed by $\sigma(Q)$.*

Let us review now the possible results of matching a query concept against a description.

Obviously, the match may fail because there is no substitution σ such that σ(Q) subsumes D. But it may also fail according to the above definition because there is no such *minimal* substitution. The simplest example of this case involves individual matching: if we have the KB describing Amy as a bear with exactly one best friend, namely herself, { **Amy:BEAR, Amy→[fills bestFriend Amy], Amy→[at-most 1 bestFriend]** }, then it follows that Amy is an instance of the following sequence of descriptions: **[all bestFriend BEAR]**, **[all bestFriend [and BEAR [all bestFriend BEAR]]]**, etc., each of which is subsumed by the previous one. Therefore, for the query **[all bestFriend ?X]**, a substitution σ mapping **?X** to any one of the above descriptions provides a situation where **Amy** is an instance of σ**[all bestFriend ?X]**, but there is no most general lower-bound to this infinite descending chain.

If we want to consider only concept subsumption, the same problem arises in languages supporting role-chain equality (**same-as**), where the conjunction of **[all bestFriend BEAR]** and **[same-as (bestFriend) (bestFriend bestFriend)]]** is subsumed by every description in the above sequence.

When the match succeeds (still using Definition 1), there are once again a number of possibilities. It may succeed with a single substitution, as in the case of matches for query concepts like **[all p ?X]** or **[at-least ?n p]**. Or it may succeed with multiple substitutions, as in the case when we match **[fills friend ?x]** against **[and [fills friend Amy] [fills friend Lulu]]**, where **?x↦Amy** and **?x↦Lulu** both yield valid matches. In fact, there may be circumstances where there are an infinite number of minimal substitutions, as in the case of matching **[and [fraction-min ?n] [fraction-max ?n]]** against **[and [fraction-min** $\frac{1}{2}$**] [fraction-max** $\frac{2}{3}$**]]**, which yields as answers substitutions for ?n of all rational numbers between a half and two thirds.

In all these cases the multiple substitutions returned by the match seem equally acceptable. However, consider the match

[and BEAR YOUNG] :: [and BEAR ?X].

In this case, **?X↦YOUNG** and **?X↦[and BEAR YOUNG]** are both valid substitutions according to our definition. In fact, so is **?X↦[or YOUNG C]** for any concept **C** disjoint from **BEAR**, such as **WOLF**. The latter substitutions seem undesirable, which would seem to indicate the need for further restrictions on the substitutions to be returned by a match; in this case, one might require that each variable be assigned as specific a value as possible. However, the following example then becomes problematic: matching

[and [all width [min 3]]
[all length [max 4]]]

against

[and [all width [min ?n]]
[all length [max ?n]]]

would normally return substitutions 3 or 4 for ?n. Yet this last definition would quite arbitrarily pick the "smallest" value of the two, say 3 in this case.

Rather than further explore the intricacies of an abstract specification of matching, we will turn to a theory of normal forms for DLs. This will yield both a particular approach to dealing with the issues above, and algorithms for computing the matching substitutions. Note that several other new operations on DLs, such as least common subsumer [6] and concept difference [16], also required restriction to normal-form in order to yield sensible results. However, this does not preclude the possibility of future work on query matching in a more general setting.

3 Structural subsumption with normalized descriptions

In looking for a suitable theory of normal forms, we note that the three DL systems widely used in practice (BACK, CLASSIC, LOOM) are implemented by first preprocessing each concept so that later operations are made easy. This initial normalization phase detects inconsistencies, and makes explicit all the implied facts or descriptions by constructing a *normal form* containing the most specific forms of the different *kinds* of descriptions entailed. For example, the normal form of the description **[at-most 0 p]** might be **and({ at-most(0,p), all(p,NOTHING) })**. For subsumption, it is these specific subterms that are used in a relatively straightforward procedure, which we call "structural subsumption".

Our version of structural subsumption (based on our earlier work on least common subsumers [6, 2]) proposes that every concept constructor **k** is to be viewed as a term constructor with a single argument, which is a value from some set D partially ordered by the relationship $\preceq_D$ (written as $D^{\preceq_D}$). *Structural subsumption checking* has the property that

$\mathbf{k}(\alpha) \Longrightarrow \mathbf{k}'(\beta)$ is true iff $\mathbf{k} = \mathbf{k}'$ and $\alpha \preceq_D \beta$.

The significance of this property is that since the normal form of a concept is usually a (conjunctively interpreted) set of descriptions built with various other constructors, one can restrict oneself to comparing ele-

ments built with the same constructor, without considering interactions between different kinds of constructors. The purpose of the normal form is then to allow subsumption to be computed using only structural comparisons of the above form. To assist this task, normal forms are also required to be non-redundant (e.g., in a set of values, none subsumes another).

Let us consider some examples of constructors and their normal form supporting structural subsumption, while at the same time introducing some refinements.

Our **min** concept constructor takes an integer as an argument, with the ordering being $\geq$, since **min**(n) $\Longrightarrow$ **min**(m) iff $n \geq m$. On the other hand, **max** also takes a integer as an argument, but the ordering is $\leq$. Since $\leq$ and $\geq$ are in fact inverses over the domain $\mathcal{N}$ of numbers, we will find it essential later to recognize this explicitly by picking one of them, say $\geq$, to be the standard ordering for $\mathcal{N}$, and use a new, auxiliary term constructor **invert**, to represent explicitly the cases in which the inverted partial order is used. So **[min 2]** and **[max 3]** will have normalized form **min**(2) and **max**(**invert**(3)), with **min** and **max** being said to have type $\mathcal{N}^{\geq}$ and $Invert(\mathcal{N}^{\geq})$ respectively. Of course, the comparison rule for the **invert** operator will be **invert**(α) $\preceq_{invert}$ **invert**(β) iff $\beta \preceq \alpha$.

Some description constructors, such as **min**, take an argument that is a value from a "basic" partial order, such as integers ordered by $\geq$ ($\mathcal{N}^{\geq}$), individuals ordered by equality ($\mathcal{I}nd^{=}$), primitive concepts and roles ordered by subsumption ($\mathcal{P}rimitiveConcept^{\Longrightarrow}$, $\mathcal{P}rimitiveRole^{\Longrightarrow}$). (For convenience, we will henceforth drop the partial order superscript.)

Other constructors take a composite value as an argument. For example, the **one-of** constructor (used to denote enumerated concepts) takes a set of individuals as an argument, while **all** takes 2 arguments: the role name and the concept. Since, we want each concept constructor **k** to take a single value as an argument, we shall extend the term notation used for concepts to also describe composite values by introducing the **set** and **tuple** functors. Thus, the normal form of **[all p C]** will be **all**(**tuple**(p,C)), while the normal form of **[one-of 2 4 6]** will be **one-of**(**set**({2,4,6})).

We must then extend the ordering $\preceq$ to composite domains. In $Tuple(D_1^{\preceq_1}, D_2^{\preceq_2})$, the values from $D_1 \times D_2$ are ordered by $\preceq_{tuple}$ according to componentwise comparison: $(n_1, r_1) \preceq_{tuple} (n_2, r_2)$ if and only if $n_1 \preceq_1 n_2$ and $r_1 \preceq_2 r_2$. For example, the intuitively valid subsumption **[all children FRENCH]** $\Longrightarrow$ **[all sons EUROPEAN]** shows that the **all** constructor is contravariant on the role argument, and hence its normalized type is $Tuple(Invert(\mathcal{P}rimitiveRole), Concept)$. As a result, we get the proof

```
[all children FRENCH]
        ⟹ [all sons EUROPEAN]
                iff
all(tuple(invert(children),FRENCH))
        ⟹ all(tuple(invert(sons),EUROPEAN))
                iff
tuple(invert(children),FRENCH)
        ⪯ tuple(invert(sons),EUROPEAN)
                iff
invert(children) ⪯ invert(sons),
        and FRENCH ⪯ EUROPEAN
                iff
sons ⟹ children, and FRENCH ⟹ EUROPEAN
```

which is true, as desired.

CONCEPT	Type of Normalized Arg.
primitive A	$Setof(\mathcal{P}rimitiveConcept)$
[one-of B_1,...,B_m]	$Invert(Setof(\mathcal{I}nd))$
[min n]	$\mathcal{N}$
[max n]	$Invert(\mathcal{N})$
[all p C]	$Tuple(Invert(\mathcal{R}ole), Concept)$
[some p C]	$Tuple(\mathcal{R}ole, Concept)$
[at-least n p]	$Tuple(\mathcal{N}, \mathcal{R}ole))$
[at-most n p]	$Tuple(Invert(\mathcal{N}), Invert(\mathcal{R}ole))$
[fills p B]	$Tuple(\mathcal{R}ole, Setof(\mathcal{I}nd))$
[and C D]	$Setof$(non-redundant terms built with the above constructors)

Table 3: Normal Form for MiniDL

For sets, the natural ordering is either $\subseteq$ or $\supseteq$, with one obtainable from the other by inversion: $A \subseteq B$ iff **invert**(A) $\supseteq$ **invert**(B). For reasons to become apparent later, it is convenient to choose $\supseteq$ as the basic ordering on sets. Therefore, since we expect **[one-of 2 4]** $\Longrightarrow$ **[one-of 2 4 6]**, the concept constructor **one-of** should take a value of type $Invert(Setof(\mathcal{I}nd))$ as its argument.

The argument of the **and** constructor is a set of descriptions, and $\supseteq$ gives exactly the right formulation for **and**(**set**({**CAT**,**BROWN**})) to be subsumed by **and**(**set**({**CAT**})). However, the elements of a set can be values that are themselves ordered (e.g., descriptions), and this has to be taken into account during subsumption checking for **set**. In particular, for **and**, we also expect **and**(**set**({**CAT**,**BROWN**})) to be subsumed by **and**(**set**({**FELINE**})). Therefore, in general we need to define the $\preceq_{set}$ relationship for two sets V and W as V$\preceq_{set}$W iff for every element x of W there is an element y in V such that $y \preceq_D x$.

Table 3 summarizes the type of the normalized domains associated with each of the concept constructors in our example DL.

The following "structural $\preceq$" algorithm embodies the above ideas; it computes structural subsumption for descriptions, as well as the $\preceq_{set}$, $\preceq_{tuple}$ and $\preceq_{invert}$ relationships, relying on the $\preceq_B$ functions provided by basic types B:

```
function structural⪯(α, β) {
    if  α and β are  values in  basic type B
          (i.e., 𝒩,ℐnd,𝒫rimitiveConcept,...) with
          partial order compare_B
      then compare_B(α,β)
    elseif α=k(α') & β=k(β') where k is a
          concept constructor
      then structural⪯(α',β')
    elseif α=invert(α') & β=invert(β')
      then structural⪯(β',α')
    elseif α=tuple(α1,α2) & β=tuple(β1,β2)
      then structural⪯(α1,β1) & structural⪯(α2,β2)
    elseif α=set(α') & β=set(β')
      then for every e in β'  choose f in α'
                                where structural⪯(f,e)
    else fail }
```

In the above, **choose x in S where P** is a nondeterministic operation, which fails if there is no value in S satisfying property P.

Note that for some DLs there may be no normal form yielding structural subsumption that is sound and complete with respect to semantic subsumption, or such a normal form may be very expensive to compute. For this reason, sometimes the structural subsumption relationship is incomplete with respect to the semantic subsumption relationship.

We will henceforth consider the case when both the concepts and queries are in normal form, and in fact define subsumption to be the relation computed between descriptions by structural$\preceq$. Therefore structural$\preceq$ computes D::Q in the case where Q has no variables.

4 Direct matching with normal forms

Having defined structural subsumption for descriptions and the concomitant normal form, we are now ready to consider the extensions necessary to match query concepts that have variables occurring in them. We obtain the match(α,β) procedure to be used for this task from the structural subsumption procedure structural$\preceq$(α,β) by making several changes.

First, if one of the arguments of match is a variable, then it is bound to the other argument to yield a substitution.

Second, we need to combine the substitutions returned by the recursive calls. For this, we shall use a function combine, which in this initial simple case requires that the substitutions agree exactly on all variables on which both are defined. Thus, combine($\sigma 1$,$\sigma 2$) signals failure if there is some variable ?x such that $\sigma 1(?x) \neq \sigma 2(?x)$, and otherwise returns $\sigma 1 \cup \sigma 2$.[2]

[2]We consider a substitution as a partial function from variables to values, which can then be viewed as a set of 2-tuples.

One additional complication arises because, according to Definition 1, we need to return the most specific substitution realizing a match. To illustrate this point, consider the concept constructor **some**, which intuitively provides typed existential quantification: **[some friend TALL]** denotes objects that have at least one filler for the **friend** role that is an instance of **TALL**. Consider the case of matching the query Q=**[some friend [at-least ?n cars]]** against the conjunction of **[some friend [and TALL [at-least 3 cars]]]** and **[some friend [and SHORT [at-least 5 cars]]]**. Although both the substitutions σ_1:?n$\mapsto$5 and σ_2:?n$\mapsto$3 have the property that σ(Q) subsumes the concept to be matched, only σ_1 should be returned by match because **[some friend [at-least 5 cars]]** is itself strictly subsumed by **[some friend [at-least 3 cars]]**.

This difficulty needs to be handled in the matching of sets: Originally, if **set**(α') is to be matched by **set**(β'), we would need to find for every e in β' some f in α' so that e matches f; however, if e contains variables, we need to make sure that e matches one of the elements of α which yields *the most specific* result. Similarly, if it is f that has variables (because of an intervening occurrence of **invert**), the match needs to return the most general possible value, so that the inversion will yield the most specific value later.[3]

As a result, we get the following algorithm for matching query concepts against descriptions:

```
function match(α, β)  {
  ;; returns a matching substitution or signals failure
   if (β is a variable)
     then return substitution { β↦α }
   elseif (α is a variable)
     then return substitution { α↦β }
   elseif α and β are  values in  basic type B
       (i.e., 𝒩,ℐnd,𝒫rimitiveConcept,...) with partial
        order compare_B
     then if compare_B(α,β)
            then return empty substitution
            else fail
   elseif α=k(α') & β=k(β') where k is a constructor
     then match(α',β')
   elseif α=invert(α') & β=invert(β')
     then match(β',α')
   elseif α=tuple(α1,α2) & β=tuple(β1,β2)
     then return combine(match(α1,β1),match(α2,β2))
   elseif α=set(α') & β=set(β')
     then{result := empty substitution;
           for every e in β'  {
              choose f in α' where σ:=match(f,e)
```

[3]Note, that only one argument of match may have variables, since originally the first parameter must be a (variable-free) concept, and **invert** only swaps the two arguments of match.

```
                succeeds and σ(e) is minimal
                and σ(f) is maximal;
            result := combine(result,σ)}
        return result; }
else fail }
```

The algorithm just presented is sound, in the sense that

Theorem 1 *If* match*(D,Q) returns substitution* σ*, then* σ *is a substitution that is a witness that D::Q succeeds according to Definition 1.*

Note that the match procedure collapses to structural subsumption checking when the second argument has no variables; in this case match(α,β) returns the empty substitution if β subsumes α, and fails otherwise. Also, in order to guarantee the minimality of the substitution, the proof of this theorem requires that the normal form of both the concept and the query be non-redundant, in the sense that in any set, no distinct pair of elements match each other.

As we shall see later, the algorithm is not complete. Moreover, the non-determinism present in the **choose** operation, and the possible relationships between choices at difference points, can cause complexity problems:

Theorem 2 *Consider the DL with concept constructors* **and**, **all**, *and primitive concepts. The problem of determining whether* match*(D,Q) succeeds in this language is NP-complete. In contrast, both subsumption and structural subsumption are polynomial time.*

The proof encodes propositional satisfiability into the choice of matches for role variables and primitive concept variables.

The following theorem indicates several cases where problems do not arise:

Theorem 3 *The complexity of checking that matching succeeds is proportional to the complexity of computing structural subsumption with normalized concepts (1) when no variable occurs more than once in the query; (2) when the* **choose** *operation always yields at most one successful result.*

The first case is relevant since it corresponds to what is known as "matching" with ordinary terms, which is a linear time operation used in programming languages such as SML.

5 Semantic matching with normal form

In the preceding sections we have defined structural subsumption and the normal form for descriptions that supports it, and then modified this subsumption algorithm to find matches when a query concept is passed to it as an argument. The result resembles matching of standard first-order terms except that we use subsumption instead of identity in comparing constants, and we treat the **set** construct specially, because we have commutative/associative operators such as conjunction. We now proceed to make matching closer to the ideal specified in Definition 1.

To illustrate the problem, observe that the previous algorithm fails to match **[and [all p ?X] [all q ?X]]** against **[and [all p DOG] [all q CAT]]** even though the substitution **?X↦ANIMAL** would yield a subsuming concept. The solution is to relax the combination of substitutions: if $\sigma1$(?X)=**DOG**, $\sigma2$(?X)=**CAT**, then combine($\sigma1$,$\sigma2$)(?X) should be **ANIMAL**, or in fact any least common subsumer of **CAT** and **DOG**, such as **[and MAMMAL HOUSE-PET]**.

To find such substitutions, note that match(a,?x) succeeds whenever ?x is some value *greater* than a, while match(?x,b) succeeds whenever ?x is bound to some value less than b. Since a variable may appear in several places, there may be multiple restrictions on it that act as lower or upper bounds. Hence our algorithm will associate with a variable a pair (A, B), representing the set of possible values $\{ f \mid a_i \preceq f$ *for each* $a_i \in A$, $f \preceq b$ *for each* $b_j \in B \}$ that could be substituted for it.

Therefore the first two statements of match(α,β) are refined as follows:

```
if (β is a variable)
    then return β↦({ α },{ })
elseif (α is a variable)
    then return α↦({ },{ β })
```

As a result, match(**min**(3),**min**(?n)) would call match(3,?n), which would return ?n↦({ 3 },{ }).

In turn, combine is modified to behave as follows: if σ_1(?**x**) = (A_1, B_1) and σ_2(?**x**) = (A_2, B_2) then combine(σ_1, σ_2)(?**x**) = $(A_1 \cup A_2,\ B_1 \cup B_2)$.

Finally, we need to extract the required substitutions from the pair of sets returned by the modified match. If match returned **?x ↦(A,{ })**, we an answer substitution that maps ?x to any least upper bound of the set A. (If no such value exists, then failure is signaled.) Similarly, for **?x↦({ },B)** we return any greatest lower bound of the set B. If the result of match maps ?x to (A, B), where neither A nor B are empty, then we return all substitutions **?x↦f**, where **f** is greater than any value in A, and less than any value in B.

Therefore, if we match the query concept **set**({ **min**(?n), **max**(**invert**(?n)) }) against **set**({ **min**(2), **max**(**invert**(4)) }), we get the answer ?n↦({ 2 }, { 4 }). This permits values between 2 and 4 inclusive to be substituted for ?n. Matching the same pattern against **set**({ **min**(4), **max**(**invert**(2)) }) fails

because ({ 4 }, { 2 }) is not satisfied by any value.

A simple special case in which the above conditions can be effectively dealt with is when all basic domains are lattices, because in this case one must only keep the least upper bound of A, and the greatest lower bound of B.

Theorem 4 *If all basic domains are either* lattices *(i.e., least upper bound and greatest lower bound are unique) or sets ordered by the identity relation, then the enhanced* match*(*D,Q*) succeeds only if* D::Q *according to Definition 1, when* C *and* Q *are normalized.*

6 Enhancements

Our experience with using query concepts shows that the query language needs to be more expressive than the language of stating facts and definitions in the knowledge base. We briefly sketch several extensions that we have found useful. (For more details, see [14].)

First, we allow restrictions on variable bindings via a nested **as** clause. For example

Sally:[at-least ?n [as [max 4]] friends]

will match only if **Sally**'s lower-bound on the number of **friends** is less than four. This is equivalent to a conjunction of matches

Sally:[at-least ?n friends] & ?n:[max 4]

but is syntactically preferable since it gives the query a structure resembling that of the knowledge base itself. As another example, the following query concept

```
[and [fills friends ?x
                    as [and BEAR [fills age ?y]]]
     [all friends ?Z as SMALL] ]
```

when matched against an individual, say **Sally** looks for friends of **Sally** who are known to be **BEARs**, and also asks for their **ages**. It also retrieves the most specific restriction on the **friends** slot which is itself, subsumed by **SMALL**. We found this kind of nested **as** clause particularly useful when printing or explaining aspects of role fillers. For example when generating a parts list for a stereo system configuration in [15], we wanted to obtain the price information on the individual stereo components. The simple pattern that should be matched against stereo systems is:

[**fills** component ?x **as** [**fills** price ?y]].

As illustrated above, we allow a qualifying query Q to apply to variables that matched a concept, as in **[all friends ?Z as Q]**, and, to be consistent, the value bound to ?Z should be viewed as an individual instance of some (meta)class described by Q. For this reason, and following a standard object-centered policy, we will think of descriptions as meta-individuals in their own right, which have their own roles and may belong to their own meta-classes. Among others, we will posit for each meta-individual a number of attributes concerning the IsA hierarchy, such as **strictlySubsumedBy** , **immediatelySubsumedBy** , etc., as well as meta-classes such as **INTERESTING-CONCEPTS**, **DATABASE-ACCESS-ROLES**, and **NAMED-DESCRIPTIONS**. In this way, if we want to see the value-restriction on the **eats** role only if it is more specific than the generic **FOOD** concept, we match against the query **[all eats ?C as [fills strictlySubsumedBy FOOD]]**.

The above mechanism can of course be used to retrieve aspects of the IsA hierarchy. For example, the "parent concepts" of class **PICKY** can be obtained by the query

PICKY :: [fills immediatelySubsumedBy ?X].

Following [7, 17], a different way of enhancing the query language is to use a richer set of concept and role constructors. Among others, in those FR-KRSs that make the open-world assumption, it is possible for the system to know that **Sally** has at least three **friends**, while only knowing the identity of two of them. For this reason, we include epistemic constructs [9] in the language. (A simple way to do this is to include an epistemic version of each constructor: **[at-least-k ?n p]** would query how many fillers are currently known for p. Our method, discussed in [14], includes **known** as a role constructor and then extends the grammar appropriately.) In addition, as a result of our experiences with CLASSIC, we believe it is imperative to include some form of extensibility in the language. For this reason we propose a **testMatch** constructor, which takes as arguments a function for matching and a list of variables to bind: **[testMatch compute-and-bind-oldest-age-of-friends '(?x)]**.

7 Related Work

We were inspired to use variables ?x in a language dealing with description logics by work in databases, as well as LOOM's FOL assertion/query language, and by other papers [8, 11] that have explored the combination of variables and frames in the context of Horn logic. As mentioned earlier, Lenzerini et al [7, 9] considered the problem of querying DL KBs using other concepts as queries, and showed various ways of using more expressive concept languages for asking questions. However, in all these cases variables were only ranging over individuals, with roles and concepts being used as unary and binary predicates. It was thus not possible to query parts of the frames themselves, which was the most significant aspect of our initial motivation — to reduce the amount of material being displayed.

In the database area, object-oriented query languages such as XSQL [12] have notation which facilitates the querying of path-expressions and allows variables for attributes as well as values: **PERSON X WHERE X.Residence[Y].City['New York'] AND X.R.City[W]**. Our work extends this through the use of subsumption as part of query matching, and the much more general nature of

frames.

Finally, our earlier work on least common subsumers for DLs [6] also used the notion of structural subsumption. The present work considerably elaborates the theory of normal forms presented in that paper. The alternative theory of normal forms for structural subsumption introduced in [16] might also have provided a good foundation for concept matching. Unfortunately, that theory relies on the notion of "clausal description" — one which has the property that if A≡**and**(A',B) then either B is A or B is the universal concept. This does not work for the **one-of** constructor of CLASSIC, since **[one-of 1 2]= [and [one-of 1 2 3] [one-of 1 2 4]]** , thus, there would be no normal form for **one-of**.

8 Desiderata Revisited

We have introduced a declarative language for inquiring about information in FR-KRSs which support notions such as roles or slots, inheritance, subsumption, etc. As desired, the query language is derived from the standard language for telling information to the KB, mostly by introducing variables for all user-specified identifiers. This query (or ASK) language can be used to "decompose" a description or frame into its various components in a uniform, declarative manner rather than using a long list of procedures. For example, **PERSON::[all friends ?X]** accesses the **all**-restriction on the **friends** role of **PERSON**, in a way that was formerly possible only using function calls such as **get-all-restriction(PERSON,friends)**.

Returning to our initial motivation, query concepts can be used to set up "masks" that work as filters and provide only the interesting information for display. In our implemented system this is done by attaching a set of query concepts to a frame[4] so that only portions matched by *some* query concept are shown or explained. Thus, we can avoid obvious information about what restrictions apply to **livingParents** by using the query concepts **[all livingParents ?X as [fills strictlySubsumedBy PERSON]]** and **[at-most ?n [as [max 1]] livingParents]**; the former succeeds only if the type constraint on **livingParents** is more specialized than **PERSON**, while the latter succeeds only if there are fewer than 2 parents.

Similarly, although the query concept describing the aspects to be printed for **FOOD** would not have a part matching the role **sugarContent**, we can require to see this for diabetics by associating **[all eats [fills sugarContent ?x]]** with the class **DIABETIC**.

We claim that the approach is applicable in general to FR-KRSs. For example, the KRL [1] frame specified as

```
PATIENT  unit specialization
  <self (a  PERSON )>
  <treatedBy (a DOCTOR with locations= (item NJ))>
```

can be queried by

```
< treatedBy ( a ?X  with ?r = ( item  ?v))>
```

Additionally, a KEE [10] frame of the form

```
PATIENT with slot treatedBy
 [ValueClass=DOCTOR,
  Cardinality.Min=2,
  Values=Kildare]
```

would be queried by

```
slot treatedBy [ValueClass=?X].
```

We have presented the initial exploratory steps concerning the notion of query concepts. Several other significant questions need to be addressed, including the treatment of the **same-as** construct in structural subsumption, the normalization of query concepts themselves, and the process of answering queries in DLs that do not use structural subsumption. One intriguing idea is to think of a query concept Q as a meta-concept with denotation $\mathcal{D}(Q)=\{\ D \mid \text{match}(D,Q)$ succeeds $\}$. This permits us to organize query concepts in a subsumption hierarchy defined as $Q1 \Longrightarrow Q2$ iff $\mathcal{D}(Q1) \subseteq \mathcal{D}(Q2)$. One reason this is interesting is because the subsumption hierarchy of ordinary queries has been found useful in the past (e.g., [5]). Moreover, this notion may be related to matches of queries against queries, Q1::Q2, which resembles the notion of unification extended with subsumption reasoning.

In conclusion, in addition to having proposed and successfully used a mechanism of clear benefit to FR-KRSs, we hope that the motivation and subtleties of query concept semantics and matching presented in this paper may lead to new directions of research for DLs; among others, we point out that query concepts are the natural counterparts of polymorphic types in programming languages, which have been extensively studied.

Acknowledgment

We are very grateful to R. Brachman, W. Cohen, H. Hirsh, H. Levesque, L. Padgham, and P. Patel Schneider for their comments on earlier versions of this paper. Charles Isbell has provided highly valued implementation assistance for the procedural version of "print masks". This research was supported in part by NSF Grant IRI-9119310.

References

[1] D. Bobrow, T. Winograd, "KRL: another perspective", *Cognitive Science*, 3(1), pp.29-42, 1979.

[4] There is inheritance and combination of such query concept filters. Also, different query concepts may be used for explanation than those used for printing objects.

[2] A.Borgida, 'Structural Subsumption: What is it and why is it important?',*1992 AAAI Fall Symposium: Issues in Description Logics*, pp.14-18.

[3] A. Borgida, R. J. Brachman, D.L McGuinness, and L. Alperin Resnick. CLASSIC: A Structural Data Model for Objects. In *Proc. SIGMOD'89*, Portland, Oregon, June 1989, pp. 59–67.

[4] R. Brachman, P.Selfridge, L.Terveen, B.Altman, A. Borgida, F. Halper, T.Kirk, A.Lazar, D.McGuinness, L.Resnick, "Knowledge representation support for data archaeology", *Int. J. of Intelligent and Cooperative Information Systems 2(2)*, June 1993, pp.159–186.

[5] M.Buchheit, M. Jeusfeld, W. Nutt, and M. Staudt, "Subsumption between queries in object-oriented databases", *Information Systems* 19(1), pp.33-54, 1994.

[6] W. Cohen, A. Borgida, and H. Hirsh, "Computing least common subsumers in description logics", *Proc. of AAAI'92*, San Jose, CA., May 1992.

[7] M. Lenzerini, A. Schaerf, 'Concept Languages as Query Languages', *Proc. AAAI'91*, pp 471-476, 1991.

[8] Donini, F., Lenzerini, M., Nardi, D., Schaerf, A. "A hybrid system integrating Datalog and concept languages", *AAAI Fall Symp. on Principles of Hybrid Reasoning*, 1991.

[9] Donini, F., Lenzerini, M., Nardi, D., Nutt, W., Schaerf, A., 'Queries, Rules and Definitions as Epistemic Sentences in Concept Languages', *Foundations of Knowledge Representation and Reasoning*, Springer LNAI 810, pp 111-132, 1994.

[10] R. Fikes, T. Kehler, "The role of frame-based representation in reasoning", *CACM* 28(9), September 1985, pp.904–920.

[11] Levy, A., Rousset., M.-C., 'CARIN: A Representation Language Combining Horn Rules and Description Logics', *Proceedings of the International Workshop on Description Logics - DL-95*, pp 44-51, Roma, Italy, 1995

[12] M. Kifer, Won Kim, Y. Sagiv: Querying Object-Oriented Databases.*Proc. SIGMOD'92*, pp.393-402.

[13] R. M. MacGregor and R. Bates. The LOOM Knowledge Representation Language. Technical Report ISI/RS-87-188, USC/Information Sciences Institute, Marina del Ray, CA, 1987.

[14] D. L. McGuinness. Explanation in Description Logics. Rutgers University, PhD thesis, 1996.

[15] D. L. McGuinness, L. Alperin Resnick, and C. Isbell. Description Logic in Practice: A CLASSIC Application. In *Proc. IJCAI'95*, Montreal, 1995.

[16] G. Teege, "Making the Difference: A Subtraction Operation for Description Logics." *Proc. KR-94*, Bonn, Germany, (May, 1994), pp.540-550.

[17] Patel-Schneider, P. F., Brachman, R. J., and Levesque, H. J., "ARGON: Knowledge representation meets information retrieval", in: *Proceedings First Conference on Artificial Intelligence Applications*, Denver, Colorado (1984) 280–286.

[18] L. Alperin Resnick, A. Borgida, R.J. Brachman, D.L. McGuinness, P.F. Patel-Schneider, C. Isbell, and K.Zalondek. CLASSIC description and reference manual for the Common Lisp implementation: Version 2.3. AI Principles Research Department, AT&T Bell Laboratories. 1995.

[19] Wright, J. R., Weixelbaum, E. S., Brown, K., Vesonder, G. T., Palmer, S. R., Berman, J. I., Moore, H. H., A knowledge-based configurator that supports sales, engineering, and manufacturing at AT&T Network Systems. In *Proceedings of the Innovative Applications of Artificial Intelligence Conference*, pp.183–193, 1993. A version of this paper also appears in *AI Magazine*, 1993, pp. 69-80.

Complexity Measures

Tractable Subclasses of the Point-Interval Algebra: A Complete Classification

Peter Jonsson, Thomas Drakengren and Christer Bäckström
Department of Computer and Information Science
Linköping University, S-581 83 Linköping, Sweden
email: {petej,thodr,cba}@ida.liu.se

Abstract

Several algebras have been proposed for reasoning about qualitative constraints over time. One of these algebras is Vilain's point-interval algebra, which can relate time points with time intervals. Apart from being a stand-alone qualitative algebra, it is also used as a subalgebra in Meiri's approach to temporal reasoning, which combines reasoning about quantitative and qualitative temporal constraints over both time points and time intervals. While the satsifiability problem for the full point-interval algebra is known to be NP-complete, not much has been known about its 4294967296 subclasses. We provide in this paper a complete classification of satisfiability for all these subclasses into polynomial and NP-complete respectively. We also identify all maximal tractable subalgebras—nine in total.

1 INTRODUCTION

Reasoning about temporal constraints is an important task in many areas of AI and elsewehere, such as planning (Allen, 1991), natural language processing (Song and Cohen, 1988), time serialization in archeology (Golumbic and Shamir, 1993) *etc.* In most applications, knowledge of temporal constraints is expressed in terms of collections of relations between time intervals or time points. Often we are only interested in qualitiative relations, *i.e.* the relative ordering of time points but not their exact occurrences in time. There are two archetypical examples of qualitative temporal reasoning: *Allen's algebra* ($\mathcal{A}$) (Allen, 1983) for reasoning about time intervals and the *point algebra* (PA) (Vilain, 1982) for reasoning about time points.

Attempts have been made to integrate reasoning about time intervals and time points. Meiri's (1991) approach to temporal reasoning makes it possible to reason about time points and time intervals with respect to both qualitative and quantitative time. This framework can be restricted to qualitiative time and the resulting fragment is known as the *qualitative algebra* (QA). In QA, a qualitative constraint between two objects O_i and O_j (each may be a point or an interval), is a disjunction of the form $(O_i r_1 O_j) \vee \ldots \vee (O_i r_k O_j)$ where each one of the r_i's is a *basic relation* that may exist between two objects. There are three types of basic relations:

1. Interval-interval relations that can hold between pairs of intervals. These relations correspond to Allen's algebra.
2. Point-point relations that can hold between pairs of points. These relations correspond to the point algebra.
3. Point-interval and interval-point relations that can hold between a point and an interval and vice-versa. These relations were introduced by Vilain (1982). The point-interval and interval-point relations are symmetric so we will only consider the point-interval relations in the sequel.

The satisfiability problem for the point algebra is known to be tractable (Vilain et al., 1989) and the satisfiability problem for Allen's algebra is NP-complete (Vilain et al., 1989). However, a large number of tractable subclasses of Allen's algebra has been reported in the literature (van Beek and Cohen, 1990; Golumbic and Shamir, 1993; Nebel and Bürckert, 1995; Drakengren and Jonsson, 1996a). Clearly, QA suffers from computational difficulties since it subsumes the Allen algebra. Even worse, Meiri (1991) shows that the satisfiability problem is NP-complete even for point-interval relations. Besides this negative result, not very much is known about the computational properties of subclasses of the point-interval algebra. This is an unfortunate situation if we want to find tractable subclasses of the qualitative algebra since the point-interval and interval-point algebras are the glue that ties the world of time points together with the world of time intervals.

We also have reasons to believe that the point-interval algebra itself can be interesting in applications such as reasoning about action and change. In certain approaches to action and change, such as the Features and Fluents framework by Sandewall (1994), a clear distinction is made between observations and actions. Typically, observations occur at a single time point while actions occur over extended periods of time. Determining temporal relations between observations and actions in a given scenario seems to be a problem which can be addressed by reasoning in the point-interval algebra.

The main result of this paper is a complete classification of all subclasses of the point-interval algebra with respect to tractability. The classification makes it possible to determine whether a given subclass is tractable or not by a simple test that can be easily carried out by hand or automatically. We have thus gained a clear picture of the borderline between tractability and intractability in the point-interval algebra. In this process, we have also taken a small step towards a deeper understanding of the qualitative algebra.

A few words on methodology seem appropriate at this point. The proof of the main theorem relies on a quite extensive case analysis performed by a computer. The number of cases considered in this analysis was approximately 10^6. Naturally, such an analysis cannot be reproduced in a research paper or be verified manually. To allow for the verification of our results, we include a description of the program used in the analysis. Furthermore, the programs used can be obtained from the authors.

The rest of this paper is organized as follows: Section 2 defines the point-interval algebra and some auxiliary concepts. Section 3 contains the classification of subclasses. Section 4 is a brief discussion of the results and Section 5 concludes the paper. Most of the proofs are postponed to the appendix. Due to space limitations, we have not been able to give all proofs in their entirety. The full proofs can be found in the technical report (Jonsson et al., 1996).

2 POINT-INTERVAL RELATIONS

The point-interval approach to reasoning about time is based on the notions *time points*, *time intervals* and *binary relations* on them. A time point p is a variable interpreted over the set of real numbers $\mathbf{R}$. A time interval I is represented by a pair $\langle I^-, I^+ \rangle$ satisfying $I^- < I^+$ where I^- and I^+ are interpreted over $\mathbf{R}$. We assume that we have a fixed universe of variable names for time points and time intervals. Then, an *$\mathcal{V}$-interpretation* is a function that maps time point variables to $\mathbf{R}$ and time interval variables to $\mathbf{R} \times \mathbf{R}$ and satisfies the previously stated restrictions. We will frequently extend the notation by denoting the the first component of $\Im(I)$ by $\Im(I^-)$ and the second by $\Im(I^+)$.

Given an interpreted time point and an interpreted time interval, their relative positions can be described by exactly one of the elements of the set $\mathbf{B}$ of five *basic point-interval relations* where each basic relation can be defined in terms of its endpoint relations (see Table 1). A formula of the form pBI where p is a time point, I a time interval and $B \in \mathbf{B}$, is said to be satisfied by an $\mathcal{V}$-interpretation iff the interpretation of the points and intervals satisfies the endpoint relations specified in Table 1.

To express indefinite information, unions of the basic relations are used, written as sets of basic relations, leading to 2^5 binary *point-interval relations*. Naturally, a set of basic relations is to be interpreted as a disjunction of the basic relations. The set of all point-interval relations $2^{\mathbf{B}}$ is denoted by $\mathcal{V}$. Relations of special interest are the *null* relation $\varnothing$ (also denoted by $\bot$) and the *universal* relation $\mathbf{B}$ (also denoted $\top$). With the notation $\neg x$ we mean the relation $\mathbf{B} - \{x\}$, *e.g.* $\neg\mathtt{a} = \{\mathtt{b}, \mathtt{s}, \mathtt{d}, \mathtt{f}\}$.

A formula of the form $p\{B_1, \ldots, B_n\}I$ is called a *point-interval* formula. Such a formula is satisfied by an $\mathcal{V}$-interpretation $\Im$ iff pB_iI is satisfied by $\Im$ for some i, $1 \leq i \leq n$. A finite set Θ of point-interval formulae is said to be *$\mathcal{V}$-satisfiable* iff there exists an $\mathcal{V}$-interpretation $\Im$ that satisfies every formula of Θ. Such a satisfying $\mathcal{V}$-interpretation is called an *$\mathcal{V}$-model* of Θ. The reasoning problem we will study is the following:

INSTANCE: A finite set Θ of point-interval formulae.
QUESTION: Does there exist an $\mathcal{V}$-model of Θ?

We denote this problem $\mathcal{V}$-SAT. In the following, we often consider restricted versions of $\mathcal{V}$-SAT where the relations used in formulae in Θ are only from a subset $\mathcal{S}$ of $\mathcal{V}$. In this case we say that Θ is a set of formulae over $\mathcal{S}$ and we use a parameter in the problem description to denote the subclass under consideration, *e.g.* $\mathcal{V}$-SAT($\mathcal{S}$).

Meiri's extended definition of the point-interval algebra consists of $\mathcal{V}$ equipped with two binary operations *intersection* and *composition*. However, this definition does not constitute an algebra because it is not closed under composition. We replace the composition operation with an operation on $\mathcal{V}$ we call *cross-composition*. The reason for introducing the algebra is that it is needed for the introduction of a closure operation which will simplify the forthcoming proofs.

Definition 2.1 Let $\mathbf{B} = \{\mathtt{b}, \mathtt{s}, \mathtt{d}, \mathtt{f}, \mathtt{a}\}$. The *point-interval algebra* consists of the set $\mathcal{V} = 2^{\mathbf{B}}$ and the operations binary *intersection* (denoted by $\cap$) and ternary *cross-composition* (denoted by $\otimes$). Intersection is defined as $\forall p, I : p(R \cap S)I \Leftrightarrow pRI \wedge pSI$ while cross-composition is defined as $\forall p, I : p(R \otimes S \otimes T)I \Leftrightarrow \exists q, J : (qRJ \wedge qSI \wedge pTJ)$.

Table 1: The five basic relations of the $\mathcal{V}$-algebra. The endpoint relation $I^- < I^+$ that is valid for all relations has been omitted.

Basic relation	Symbol	Example	Endpoint rel.
p before I	b	p III	$p < I^-$
p starts I	s	p III	$p = I^-$
p during I	d	p III	$I^- < p < I^+$
p finishes I	f	p III	$p = I^+$
p after I	a	p III	$p > I^+$

It can easily be verified that $R \otimes S \otimes T = \bigcup\{B \otimes B' \otimes B'' | B \in R, B'' \in S, B''' \in T\}$, *i.e.* cross-composition is the union of the component-wise cross-composition of basic relations.

Next, we introduce a *closure* operation $\mathcal{C}_\mathcal{V}$ together with a *duality* operator $\mathcal{D}_\mathcal{V}$. Both $\mathcal{C}_\mathcal{V}$ and $\mathcal{D}_\mathcal{V}$ transform a given subclass of $\mathcal{V}$ to one that is polynomially equivalent to the original subclass wrt. satisfiability. The closure operation is similar to the closure operation for the Allen algebra introduced in (Nebel and Bürckert, 1995), and is defined as follows.

Definition 2.2 Let $\mathcal{S} \subseteq \mathcal{V}$. Then we denote by $\mathcal{C}_\mathcal{V}(\mathcal{S})$ the *$\mathcal{V}$-closure* of $\mathcal{S}$, defined as the least subalgebra containing $\mathcal{S}$ closed under intersection and cross-composition.

A program for computing $\mathcal{V}$-closures can be obtained from the authors.

Lemma 2.3 Let $\mathcal{S} \subseteq \mathcal{V}$. Then $\mathcal{V}$-SAT($\mathcal{C}_\mathcal{V}(\mathcal{S})$) can be polynomially transformed to $\mathcal{V}$-SAT(S).

Corollary 2.4 Let $\mathcal{S} \subseteq \mathcal{V}$. $\mathcal{V}$-SAT($\mathcal{S}$) is polynomial iff $\mathcal{V}$-SAT($\mathcal{C}_\mathcal{V}(\mathcal{S})$) is polynomial. $\mathcal{V}$-SAT($\mathcal{S}$) is NP-complete iff $\mathcal{V}$-SAT($\mathcal{C}_\mathcal{V}(\mathcal{S})$) is NP-complete.

Next we introduce the duality operator and show that it has the same transformational properties as the closure operation.

Definition 2.5 Let $R \in \mathcal{V}$. Define $\mathcal{D}_\mathcal{V}(R)$ as the set $\{\beta(r) \mid r \in R\}$ where $\beta(r)$ is defined as follows: $\beta(\mathtt{b}) = \mathtt{a}$, $\beta(\mathtt{s}) = \mathtt{f}$, $\beta(\mathtt{d}) = \mathtt{d}$, $\beta(\mathtt{f}) = \mathtt{s}$ and $\beta(\mathtt{a}) = \mathtt{b}$.

Let $\mathcal{S} \subseteq \mathcal{V}$. Define $\mathcal{D}_\mathcal{V}(\mathcal{S})$ as the set $\{\mathcal{D}_\mathcal{V}(R) \mid R \in \mathcal{S}\}$.

Lemma 2.6 Let $\mathcal{S} \subseteq \mathcal{V}$. Then $\mathcal{V}$-SAT($\mathcal{D}_\mathcal{V}(\mathcal{S})$) can be polynomially transformed to $\mathcal{V}$-SAT(S).

Proof sketch: Let Θ be an instance of the $\mathcal{V}$-SAT($\mathcal{D}_\mathcal{V}(\mathcal{S})$) problem that have a $\mathcal{V}$-model $\Im$. Construct the following $\mathcal{V}$-SAT(S) instance: $\Theta' = \{p\mathcal{D}_\mathcal{V}(R)I \mid pRI \in \Theta\}$. A $\mathcal{V}$-interpretation $\Im'$ of Θ' can be constructed as follows: Let $\Im'(p) = -\Im(p)$ for each time point p appearing in Θ and let $\Im'(I^-) = -\Im(I^+)$, $\Im'(I^+) = -\Im(I^-)$ for each time interval I appearing in Θ. Clearly, $\Im'$ is a $\mathcal{V}$-model of Θ'. Showing the converse direction is analogous. □

Corollary 2.7 Let $\mathcal{S} \subseteq \mathcal{V}$. $\mathcal{V}$-SAT($\mathcal{S}$) is polynomial iff $\mathcal{V}$-SAT($\mathcal{D}_\mathcal{V}(\mathcal{S})$) is polynomial. $\mathcal{V}$-SAT($\mathcal{S}$) is NP-complete iff $\mathcal{V}$-SAT($\mathcal{D}_\mathcal{V}(\mathcal{S})$) is NP-complete.

3 CLASSIFICATION OF $\mathcal{V}$

We begin this section by defining nine tractable subalgebras of the point-interval algebra. Later on, we show that these algebras are the only maximal tractable subalgebras of $\mathcal{V}$. Before we can define the algebras we need a definition concerning the point algebra.

Definition 3.1 A *PA formula* is an expression of the form xry where r is a member of $\{<, \leq, =, \neq, \geq, >, \bot, \top\}$ and x, y denote real-valued variables. The symbol $\bot$ denotes the relation $\varnothing$ which is unsatisfiable for every choice of $x, y \in \mathbf{R}$. Similarly, $\top$ denotes the relation $\mathbf{R} \times \mathbf{R}$ which is satisfiable for every choice of $x, y \in \mathbf{R}$.

Let Ω be a set of PA formulae and X the set of variables appearing in Ω. An assignment of real values to the variables in X is said to be an *PA-interpretation* of Ω. Furthermore, Ω is *satisfiable* iff there exists an PA-interpretation $\Im$ such that for each formula $xry \in \Omega$, $\Im(x)r\Im(y)$ holds. Such an PA-interpretation $\Im$ is said to be an *PA-model* of Ω.

The first algebra we will consider has a very close connection to PA. It is defined as follows.

Definition 3.2 The set $\mathcal{V}^{23}$ consists of those relations in $\mathcal{V}$ that can be expressed as one or more PA formulae over time points and endpoints of intervals.

The other eight subalgebras are defined in terms of the $\mathcal{C}_\mathcal{V}$ and $\mathcal{D}_\mathcal{V}$ operators.

Definition 3.3

$$v_{\mathtt{s}}^{20} = \{\{\mathtt{s}\}, \{\mathtt{b}, \mathtt{s}\}, \{\mathtt{b}, \mathtt{a}\}, \neg\mathtt{d}, \neg\mathtt{f}\}, \mathcal{V}_{\mathtt{s}}^{20} = \mathcal{C}_\mathcal{V}(v_{\mathtt{s}}^{20})$$

$$\mathcal{V}_{\mathtt{f}}^{20} = \mathcal{D}_\mathcal{V}(\mathcal{V}_{\mathtt{s}}^{20})$$

$$v_{\mathtt{d}}^{20} = \{\{\mathtt{b}, \mathtt{a}\}, \neg\mathtt{b}, \neg\mathtt{s}, \neg\mathtt{f}, \neg\mathtt{a}\}, \mathcal{V}_{\mathtt{d}}^{20} = \mathcal{C}_\mathcal{V}(v_{\mathtt{d}}^{20})$$

$$v_{\neg\mathtt{a}}^{18} = \{\{\mathtt{d}\}, \{\mathtt{b}, \mathtt{s}, \mathtt{a}\}, \neg\mathtt{s}, \neg\mathtt{f}, \neg\mathtt{a}\}, \mathcal{V}_{\neg\mathtt{a}}^{18} = \mathcal{C}_\mathcal{V}(v_{\neg\mathtt{a}}^{18})$$

$$\mathcal{V}_{\neg\mathtt{b}}^{18} = \mathcal{D}_\mathcal{V}(\mathcal{V}_{\neg\mathtt{a}}^{18})$$

$$v_{\neg\mathtt{d}}^{18} = \{\{\mathtt{d}\}, \{\mathtt{b}, \mathtt{d}\}, \neg\mathtt{s}, \neg\mathtt{d}, \neg\mathtt{f}\}, \mathcal{V}_{\neg\mathtt{d}}^{18} = \mathcal{C}_\mathcal{V}(v_{\neg\mathtt{d}}^{18})$$

$$\mathcal{V}_{\mathbf{s}}^{17} = \{\perp\} \cup \{r \in \mathcal{V} \mid \{\mathbf{s}\} \subseteq r\}$$

$$\mathcal{V}_{\mathbf{f}}^{17} = \mathcal{D}_{\mathcal{V}}(\mathcal{V}_{\mathbf{s}}^{18})$$

Given a subalgebra $\mathcal{V}_y^x$, x is the number of relations in the algebra and y is an element that is unique for $\mathcal{V}_y^x$ among the subalgebras of size x. For instance, $\mathcal{V}_{\mathbf{s}}^{17}$ is the only subalgebra of size 17 that contains $\{\mathbf{s}\}$. Let $\mathcal{V}_P$ be the set of all subalgebras in Definition 3.3 plus the algebra $\mathcal{V}^{23}$. The relations included in each of these algebras can be found in Table 2. Further, let $\mathcal{V}_{NP}$ denote the set of subalgebras listed in Table 3. We have the following theorem.

Theorem 3.4 If $V \in \mathcal{V}_P$ then $\mathcal{V}$-SAT(V) is polynomial. If $V \in \mathcal{V}_{NP}$ then $\mathcal{V}$-SAT(V) is NP-complete.

Proof: See Appendices A and B for the results concerning $\mathcal{V}_P$ and $\mathcal{V}_{NP}$, respectively. □

The main theorem can now be stated.

Theorem 3.5 For $\mathcal{S} \subseteq \mathcal{V}$, $\mathcal{V}$-SAT($\mathcal{S}$) is polynomial iff $\mathcal{S}$ is a subset of some member of $\mathcal{V}_P$.

Proof: *if:* For each $V \in \mathcal{V}_P$, $\mathcal{V}$-SAT(V) is polynomial by Theorem 3.4.

only-if: Assume there exists a subclass $\mathcal{S} \subseteq \mathcal{V}$ such that $\mathcal{V}$-SAT($\mathcal{S}$) is polynomial and $\mathcal{S}$ is not a subset of any algebra in $\mathcal{V}_P$. Without loss of generality, let $\mathcal{S}$ be such a class with the least number of elements. For each subalgebra V in $\mathcal{V}_P$, choose a relation x such that $x \in \mathcal{S}$ and $x \notin V$. This can always be done since $\mathcal{S} \not\subseteq V$. Let X be the set of these relations. The following holds for X:

1. $\mathcal{V}$-SAT(X) is polynomial since $X \subseteq \mathcal{S}$.
2. X is not a subset of any algebra in $\mathcal{V}_P$.

$\mathcal{S}$ is a minimal set satisfying (1) and (2) above. Hence, $|\mathcal{S}| \leq |X|$. Furthermore, $X \subseteq \mathcal{S}$ so $|\mathcal{S}| \leq |X|$ and $|\mathcal{S}| = |X|$. The set $\mathcal{V}_P$ contains nine algebras so by the construction of X, $|X| \leq 9$. As a consequence, $|\mathcal{S}| \leq 9$.

To show that $\mathcal{S}$ does not exist, a machine-assisted case analysis of the following form was performed:

1. Generate all subsets of $\mathcal{V}$ of size ≤ 9. There are $\sum_{i=0}^{9} \binom{32}{i} \approx 4.3 \times 10^6$ such subsets.
2. Let $\mathcal{T}$ be such a set. Test: $\mathcal{T}$ is a subset of some subalgebra in $\mathcal{V}_P$ or $D \subseteq \mathcal{C}_{\mathcal{V}}(\mathcal{T})$ for some $D \in \mathcal{V}_{NP}$.

The test succeeds for all $\mathcal{T}$. Hence, by Theorem 3.4, either $\mathcal{V}$-SAT($\mathcal{S}$) is NP-complete or $\mathcal{S}$ is a subset of some member of $\mathcal{V}_P$. Both cases contradict our initial assumptions so $\mathcal{S}$ cannot exist. □

4 DISCUSSION

We have only considered qualitative relations between time points and intervals in this paper. For certain applications this is satisfactory—for others we must have the ability to reason also about quantitative time. Previous research on reasoning about combined qualitative and quantitative time has proven this problem to be computationally hard. However, recent results show that tractable reasoning is possible in certain subclasses of Allen's algebra augmented with quite advanced quantitative information. The linear-programming approach by Jonsson and Bäckström (1996) offers a straightforward method for extending the ORD-Horn subclass with quantitative information. Several other subclasses of Allen's algebra with this property are exhibited in (Drakengren and Jonsson, 1996b). Almost certainly, these methods can be adapted to the point-interval algebra. This opens up for some interesting future research. Another interesting research direction is the study of tractable subclasses of Meiri's unrestricted approach, *i.e.*, allowing for time points and time intervals to be both qualitatively and quantitatively related.

The number of subclasses of $\mathcal{V}$ ($2^{32} \approx 4.3 \times 10^9$) is very small in comparison with the $2^{8192} \approx 10^{2466}$ subclasses of $\mathcal{A}$. In principle it would have been possible to enumerate all subclasses of $\mathcal{V}$ with the aid of a computer. Obviously, this is not the case with $\mathcal{A}$ (at least not with the computers available today). If we want to classify the subclasses of $\mathcal{A}$ with respect to tractability, we must use other methods. We are not pessimistic about the possibility of creating a complexity map of $\mathcal{A}$. Similar projects have been successfully performed in mathematics and computer science. A well-known example is the proof of the four-colour theorem (Appel and Haken, 1976) which combine theoretical studies of planar graphs with extensive machine-generated case analyses. It seems likely that we shall need methods that combine theoretical studies of the structure of $\mathcal{A}$ with brute-force computer methods. Here we can see a challenge for both theoreticians and practitioners in computer science.

5 CONCLUSIONS

We have studied computational properties of the point-interval algebra. All of the 2^{32} possible subclasses are classified with respect to whether their corresponding satisfiability problem is tractable or not. The classification reveals that there are exactly nine maximally tractable subclasses of the algebra.

References

AAAI (1996). *Proc. 13th (US) Nat'l Conf. on Artif. Intell. (AAAI-96)*, Portland, OR, USA.

Table 2: The maximal tractable subalgebras of $\mathcal{V}$.

	$\mathcal{V}^{23}$	$\mathcal{V}_{\mathbf{s}}^{20}$	$\mathcal{V}_{\mathbf{f}}^{20}$	$\mathcal{V}_{\mathbf{d}}^{20}$	$\mathcal{V}_{\neg\mathbf{a}}^{18}$	$\mathcal{V}_{\neg\mathbf{b}}^{18}$	$\mathcal{V}_{\neg\mathbf{d}}^{18}$	$\mathcal{V}_{\mathbf{s}}^{17}$	$\mathcal{V}_{\mathbf{f}}^{17}$
$\perp$	•	•	•	•	•	•	•	•	•
{b}	•	•	•	•	•	•	•		
{s}	•	•						•	
{b, s}	•	•	•		•		•	•	
{d}	•			•	•	•	•		
{b, d}	•		•	•	•	•	•		
{s, d}	•			•		•		•	
{b, s, d}	•		•	•	•	•	•	•	
{f}	•		•						•
{b, f}			•						•
{s, f}								•	•
{b, s, f}			•					•	•
{d, f}	•			•	•				•
{b, d, f}	•		•	•	•				•
{s, d, f}	•			•				•	•
{b, s, d, f}	•		•	•	•			•	•
{a}	•	•	•	•	•	•	•		
{b, a}		•	•	•	•	•	•		
{s, a}		•						•	
{b, s, a}		•	•		•		•	•	
{d, a}	•	•		•	•	•	•		
{b, d, a}	•	•	•	•	•	•	•		
{s, d, a}	•	•		•		•		•	
{b, s, d, a}	•	•	•	•	•	•	•	•	
{f, a}	•	•	•			•	•		•
{b, f, a}		•	•			•	•		•
{s, f, a}		•						•	•
{b, s, f, a}		•	•				•	•	•
{d, f, a}	•	•		•	•	•	•		•
{b, d, f, a}	•	•	•	•	•	•	•		•
{s, d, f, a}	•	•		•		•		•	•
$\top$	•	•	•	•	•	•	•	•	•

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Comm. ACM*, 26(11):832–843.

Allen, J. F. (1991). Temporal reasoning and planning. In Allen, J., Kautz, H., Pelavin, R., and Tenenberg, J., editors, *Reasoning about Plans*, chapter 1, pages 1–67. Morgan Kaufmann.

Appel, K. and Haken, W. (1976). Every planar map is four colorable. *Bulletin of the American Mathematical Society*, 82:711–712.

Drakengren, T. and Jonsson, P. (1996a). Maximal tractable subclasses of Allen's interval algebra: Preliminary report. In (AAAI, 1996).

Drakengren, T. and Jonsson, P. (1996b). Eight maximal tractable subclasses of Allen's algebra with metric time. Manuscript.

Golumbic, M. C. and Shamir, R. (1993). Complexity and algorithms for reasoning about time: A graph-theoretic approach. *J. ACM*, 40(5):1108–1133.

Jonsson, P. and Bäckström, C. (1996). A linear-programming approach to temporal reasoning. In (AAAI, 1996).

Jonsson, P., Drakengren, T., and Bäckström, C. (1996). Tractable subclasses of the point-interval algebra: A complete classification. Technical report. In preparation.

Meiri, I. (1991). Combining qualitative and quantitative constraints in temporal reasoning. In *Proc. 9th (US) Nat'l Conf. on Artif. Intell. (AAAI-91)*, pages 260–267, Anaheim, CA, USA.

Nebel, B. and Bürckert, H.-J. (1993). Software for machine assisted analysis of Allen's interval algebra. Available from the authors by anonymous ftp from duck.dfki.uni-sb.de as /pub/papers/DFKI-others/RR-93-11.programs.tar.Z.

Nebel, B. and Bürckert, H.-J. (1995). Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra. *J. ACM*, 42(1):43–66.

Sandewall, E. (1994). *Features and Fluents*. Oxford University Press.

Song, F. and Cohen, R. (1988). The interpretation of temporal relations in narrative. In *Proc. 7th (US)*

Nat'l Conf. on Artif. Intell. (AAAI-88), pages 745–750, St. Paul, MN, USA.

van Beek, P. and Cohen, R. (1990). Exact and approximate reasoning about temporal relations. *Comput. Intell.*, 6(3):132–144.

Vilain, M. B. (1982). A system for reasoning about time. In *Proc. 2nd (US) Nat'l Conf. on Artif. Intell. (AAAI-82)*, pages 197–201, Pittsburgh, PA, USA.

Vilain, M. B., Kautz, H. A., and van Beek, P. G. (1989). Constraint propagation algorithms for temporal reasoning: A revised report. In *Readings in Qualitative Reasoning about Physical Systems*, pages 373–381. San Mateo, CA.

Appendix

This appendix collects the tractability and intractability proofs needed for the proof of Theorem 3.5. The former results can be found in part A and the latter results in part B. Due to space limitations, we have not been able to give all proofs in their entirety. Hence, certain proofs are only sketched or outlined. The full proofs can be found in the technical report (Jonsson et al., 1996).

A TRACTABILITY RESULTS

To make the forthcoming proofs less cumbersome, we need a few results of more general character. These can be found in Section A.1. The actual proofs of tractability can be found in Section A.2.

A.1 MODEL TRANSFORMATIONS

One of our main vehicles for showing computational properties of different subclasses is that of *model transformations*. It is a method for transforming a solution of one problem to a solution of a related problem.

Definition A.1 Let T be a mapping on $\mathcal{V}$-interpretations. We say that T is a *model transformation*.

This definition is very general. To make it applicable in practice, we need a way of describing model transformations in greater detail.

Definition A.2 Let T be a model transformation. A function $f_T : \mathbf{B} \to 2^{\mathbf{B}}$ is a *description* of T iff for arbitrary $\mathcal{V}$-interpretations $\Im$, the following holds: if $b \in \mathbf{B}$ and $p(b)J$ under $\Im$ then $p(f_T(b))J$ under $T(\Im)$. A description f_T can be extended to handle disjunctions in the obvious way: $f_T(R) = \bigcup_{r \in R} f_T(r)$.

We can now provide two lemmata showing how model transformations can be used.

Lemma A.3 Let $\mathcal{R} = \{r_1, \ldots, r_n\} \subseteq \mathcal{V}$ be such that $\mathcal{V}$-SAT($\mathcal{R}$) is polynomial. Let $\mathcal{R}' = \{r'_1, \ldots, r'_n\} \subseteq \mathcal{V}$. Let T be a model transformation and f_T a description of T. If the following holds: (1) $r_k \subseteq r'_k$ for $1 \leq k \leq n$; and (2) $f_T(r'_k) \subseteq r_k$ for $1 \leq k \leq n$, then $\mathcal{V}$-SAT($\mathcal{R}'$) is polynomial.

Proof: Let Θ' be a instance of $\mathcal{V}$-SAT($\mathcal{R}'$). Let $\Theta = \{pr_k J \mid pr'_k J \in \Theta'\}$. Obviously, this is a polynomial transformation and Θ is an instance of $\mathcal{V}$-SAT($\mathcal{R}$). We show that Θ has a model iff Θ' has a model. Since it can be decided in polynomial time whether Θ has a model or not, the lemma follows.

only-if: Let $\Im$ be a model of Θ. Recall that $r_k \subseteq r'_k$, $1 \leq k \leq n$. Hence, $\Im$ is a model for Θ' since every relation $pr'I \in \Theta'$ is weaker than the corresponding relation $prI \in \Theta$.

if: Let $\Im'$ be a model of Θ'. We show that $T(\Im')$ is a model of Θ. Arbitrarily choose a formula $pr_k I$ in Θ. Clearly, there exists a formula $pr'_k I \in \Theta'$. Thus, we have that $pr'_k I$ under $\Im'$ which implies $pf_T(r'_k)I$ under $T(\Im')$ since f_T is a description of T. Furthermore, $f_T(r'_k) \subseteq r_k$ so $pr_k I$ under $T(\Im)$. Hence, $T(\Im)$ is a model of Θ. □

Lemma A.4 Let $\mathcal{R} = \{r_1, \ldots, r_n\} \subseteq \mathcal{V}$ and $\mathcal{R}' = \{r'_1, \ldots, r'_n\} \subseteq \mathcal{V}$ be such that $\mathcal{V}$-SAT($\mathcal{R}$) is NP-complete. If there exists a model transformation T with a description f_T such that $f_T(r'_k) \subseteq r_k$ for every $1 \leq k \leq n$ then $\mathcal{V}$-SAT($\mathcal{R}'$) is NP-complete.

Proof: Let Θ be an arbitrary instance of $\mathcal{V}$-SAT($\mathcal{R}$). Let $\Theta' = \{pf_T(r)J \mid prJ \in \Theta\}$. Obviously, this is a polynomial transformation and Θ' is an instance of $\mathcal{V}$-SAT($\mathcal{R}'$). We show that Θ is satisfiable iff Θ' is satisfiable.

if: Let $\Im'$ be a model of Θ'. Recall that $f_T(r'_k) \subseteq r_k$, $1 \leq k \leq n$. Hence, $\Im'$ is a model for Θ since every relation $prI \in \Theta$ is weaker than the corresponding relation $pf_T(r)I \in \Theta'$.

only-if: Let $\Im$ be a model of Θ. We show that $T(\Im)$ is a model of Θ'. Arbitrarily choose a formula prI in Θ'. By the construction of Θ', there exists a formula $psI \in \Theta$ such that $r = f_T(s)$. Thus, we have that psI under $\Im$ which implies $pf_T(s)I$ under $T(\Im)$ since f_T is a description of T. Hence, IrJ under $T(\Im)$. □

Before we define a number of model transformations that we will use later on, we need an auxiliary definition.

Definition A.5 Let $S \subseteq \mathbf{R}$ be finite and denote the absolute value of x with abs(x) The *minimal distance in* S, MD(S), is defined as $\min\{\text{abs}(x - y) \mid x, y \in S \text{ and } x \neq y\}$.

Observe that $|S| \geq 2$ in order to make MD(S) defined. For all such S, MD(S) > 0. The definition of

minimal distance can be extended to $\mathcal{V}$-interpretations in the following way: Let $\Im$ be an $\mathcal{V}$-interpretation that assigns values to a set of time points $\mathcal{P}$ and a set of intervals $\mathcal{I}$. Let $\mathrm{MD}(\Im) = \mathrm{MD}(\{\Im(p) \mid p \in \mathcal{P}\} \cup \{\Im(I^-), \Im(I^+) \mid I \in \mathcal{I}\})$.

In the forthcoming four propositions, let $\Im$ be an arbitrary $\mathcal{V}$-interpretation and let $\varepsilon = \mathrm{MD}(\Im)$.

Proposition A.6 Define $T_1(\Im)$ as follows: (1) for each time point p let $T_1(\Im)(p) = \Im(p)$; (2) for each time interval I let $T_1(\Im)(I^-) = \Im(I^-) + \varepsilon$ and $T_1(\Im)(I^+) = \Im(I^+)$. The description of T_1, f_{T_1} is then: $f_{T_1}(\mathtt{b}) = \{\mathtt{b}\}$, $f_{T_1}(\mathtt{s}) = \{\mathtt{b}\}$, $f_{T_1}(\mathtt{d}) = \{\mathtt{d}\}$, $f_{T_1}(\mathtt{f}) = \{\mathtt{f}\}$ and $f_{T_1}(\mathtt{a}) = \{\mathtt{a}\}$.

Proposition A.7 Define $T_2(\Im)$ as follows: (1) for each time point p let $T_2(\Im)(p) = \Im(p)$; (2) for each time interval I let $T_2(\Im)(I^-) = \Im(I^-) + \varepsilon$ and $T_2(\Im)(I^+) = \Im(I^+) - \varepsilon$. The description of T_2, f_{T_2} is then: $f_{T_2}(\mathtt{b}) = \{\mathtt{b}\}$, $f_{T_2}(\mathtt{s}) = \{\mathtt{b}\}$, $f_{T_2}(\mathtt{d}) = \{\mathtt{d}\}$, $f_{T_2}(\mathtt{f}) = \{\mathtt{a}\}$ and $f_{T_2}(\mathtt{a}) = \{\mathtt{a}\}$.

Proposition A.8 Define $T_3(\Im)$ as follows: (1) for each time point p let $T_3(\Im)(p) = \Im(p)$; (2) for each time interval I let $T_3(\Im)(I^-) = \Im(I^-) - \varepsilon$ and $T_3(\Im)(I^+) = \Im(I^+)$. The description of T_3, f_{T_3} is then: $f_{T_3}(\mathtt{b}) = \{\mathtt{b}\}$, $f_{T_3}(\mathtt{s}) = \{\mathtt{d}\}$, $f_{T_3}(\mathtt{d}) = \{\mathtt{d}\}$, $f_{T_3}(\mathtt{f}) = \{\mathtt{f}\}$ and $f_{T_3}(\mathtt{a}) = \{\mathtt{a}\}$.

Proposition A.9 Define $T_4(\Im)$ as follows: (1) for each time point p let $T_4(\Im)(p) = \Im(p)$; (2) for each time interval I let $T_4(\Im)(I^-) = \Im(I^-)$ and $T_4(\Im)(I^+) = \Im(I^+) - \varepsilon$ The description of T_4, f_{T_4} is then: $f_{T_4}(\mathtt{b}) = \{\mathtt{b}\}$, $f_{T_4}(\mathtt{s}) = \{\mathtt{s}\}$, $f_{T_4}(\mathtt{d}) = \{\mathtt{d}\}$, $f_{T_4}(\mathtt{f}) = \{\mathtt{a}\}$ and $f_{T_4}(\mathtt{a}) = \{\mathtt{a}\}$.

A.2 PROOFS OF TRACTABILITY

Proving tractability of $\mathcal{V}$-SAT($\mathcal{V}^{23}$) is straightforward.

Proposition A.10 Deciding satisfiability of a set of PA formulae is polynomial.

Proof: See (Vilain et al., 1989). □

Lemma A.11 $\mathcal{V}$-SAT($\mathcal{V}^{23}$) is polynomial.

Proof: Follows immediately from the definition of $\mathcal{V}^{23}$ and the previous proposition. □

Lemma A.12 $\mathcal{V}$-SAT($\mathcal{V}_{\mathtt{s}}^{20}$) is polynomial.

Proof: By Corollary 2.4, it is sufficient to show that $\mathcal{V}$-SAT($v_{\mathtt{s}}^{20}$) is polynomial. Let Θ be an arbitrary instance of $\mathcal{V}$-SAT($v_{\mathtt{s}}^{20}$). Define the function $\sigma : v_{\mathtt{s}}^{20} \to \{\leq, =, \neq, \top\}$ as follows: $\sigma(\{\mathtt{s}\}) =$ "$=$", $\sigma(\{\mathtt{b}, \mathtt{s}\}) =$ "$\leq$", $\sigma(\{\mathtt{b}, \mathtt{a}\}) =$ "$\neq$", $\sigma(\{\mathtt{b}, \mathtt{s}, \mathtt{d}, \mathtt{a}\}) =$ "$\top$" and $\sigma(\{\mathtt{b}, \mathtt{s}, \mathtt{f}, \mathtt{a}\}) =$ "$\top$".

Let $\Theta' = \{x_p \sigma(R) y_I \mid pRI \in \Theta\}$. In the sequel, y_I will be considered as the starting point of the interval I. By Proposition A.10, it is polynomial to decide R-satisfiability of Θ'. We show that Θ is $\mathcal{V}$-satisfiable iff Θ' is R-satisfiable.

if: Let $\Im'$ be an R-model of Θ'. Let $\varepsilon = \mathrm{MD}(\Im')/2$. We construct an $\mathcal{V}$-interpretation $\Im$ of Θ as follows:

- For each variable x_p in Θ' let $\Im(p) = \Im'(x_p)$.
- For each variable y_I in Θ' let $\Im(I^-) = \Im'(y_I)$ and $\Im(I^+) = \Im'(y_I) + \varepsilon$

We show that $\Im$ is an $\mathcal{V}$-model of Θ. Let pRI be an arbitrary formula in Θ. We have five different cases.

1. $R = \{\mathtt{s}\}$. Then $\Im'(x_p)\sigma(\{\mathtt{s}\})\Im'(y_I) \Leftrightarrow \Im'(x_p) = \Im'(y_I) \Leftrightarrow \Im(p) = \Im(I^-)$. Consequently, $p\{\mathtt{s}\}I$ under $\Im$.
2. $R = \{\mathtt{b}, \mathtt{s}\}$. Then $\Im'(x_p)\sigma(\{\mathtt{b}, \mathtt{s}\})\Im'(y_I) \Leftrightarrow \Im'(x_p) \leq \Im'(y_I) \Leftrightarrow \Im(p) \leq \Im(I^-)$. Consequently, $p\{\mathtt{b}, \mathtt{s}\}I$ under $\Im$.
3. $R = \{\mathtt{b}, \mathtt{a}\}$. Then $\Im'(x_p)\sigma(\{\mathtt{b}, \mathtt{a}\})\Im'(y_I) \Leftrightarrow \Im'(x_p) \neq \Im'(y_I) \Leftrightarrow \Im(p) \neq \Im(I^-)$. If $\Im(p) < \Im(I^-)$ then $p\{\mathtt{b}\}I$ under $\Im$. If $\Im(p) > \Im(I^-)$ then $\Im(p) > \Im(I^+)$ since $I^+ = I^- + \varepsilon$. In this case $p\{\mathtt{a}\}I$ under $\Im$. Consequently, $p\{\mathtt{b}, \mathtt{a}\}I$ under $\Im$.
4. $R = \{\mathtt{b}, \mathtt{s}, \mathtt{f}, \mathtt{a}\}$. Assume $p\{\mathtt{d}\}I$ under $\Im$. Then $\Im(I^-) < \Im(p) < \Im(I^+) \Leftrightarrow \Im(I^-) < \Im(p) < \Im(I^-) + \varepsilon \Leftrightarrow \Im'(y_I) < \Im'(x_p) < \Im'(y_I) + \varepsilon$ which contradicts the choice of ε. Hence, $p\{\mathtt{b}, \mathtt{s}, \mathtt{f}, \mathtt{a}\}I$ under $\Im$
5. $R = \{\mathtt{b}, \mathtt{s}, \mathtt{d}, \mathtt{a}\}$. This case is analogous to the previous case.

only-if: Let $\Im$ be an $\mathcal{V}$-model of Θ. We construct an R-interpretation $\Im'$ of Θ' as follows:

- For each time point p in Θ let $\Im'(x_p) = \Im(p)$.
- For each time interval I in Θ let $\Im'(y_I) = \Im(I^-)$.

Next, we show that $\Im'$ is an R-model of Θ'. Let $x_p R y_I$ be an arbitrary formula in Θ'. We have four different cases:

1. $R =$ "$=$". Assume $\Im'(x_p) \neq \Im'(y_I)$. Then $\Im(p) \neq \Im(I^-)$ and $p\{\mathtt{b}, \mathtt{d}, \mathtt{f}, \mathtt{a}\}I$ under $\Im$. But $x_p = y_I \in \Theta'$ iff $p\{\mathtt{s}\}I \in \Theta$ which leads to a contradiction. Hence, $x_p = y_I$ under $\Im'$.
2. $R =$ "$\neq$". This case is analogous to the previous case.
3. $R =$ "$\leq$". Assume $\Im'(x_p) > \Im'(y_I)$. Then $\Im(p) > \Im(I^-)$ and $p\{\mathtt{d}, \mathtt{f}, \mathtt{a}\}I$ under $\Im$. But $x_p \leq y_I \in \Theta'$ iff $p\{\mathtt{b}, \mathtt{s}\}I \in \Theta$ which leads to a contradiction. Hence, $x_p \leq y_I$ under $\Im'$.
4. $R =$ "$\top$". This relation holds trivially in every R-model of Θ'. □

Lemma A.13 $\mathcal{V}$-SAT($\mathcal{V}_{\mathtt{d}}^{20}$) is polynomial.

Proof sketch: By Corollary 2.4, it is sufficient to show that $\mathcal{V}$-SAT($v_{\mathtt{d}}^{20}$) is polynomial. Let Θ be an arbitrary instance of $\mathcal{V}$-SAT($v_{\mathtt{d}}^{20}$). Define the function $\sigma : v_{\mathtt{d}}^{20} \to \{\leq, \geq, \neq, \top\}$ as follows: $\sigma(\{\mathtt{b},\mathtt{a}\}) =$ "$\neq$", $\sigma(\{\mathtt{s},\mathtt{d},\mathtt{f},\mathtt{a}\}) =$ "$\geq$", $\sigma(\{\mathtt{b},\mathtt{s},\mathtt{d},\mathtt{f}\}) =$ "$\leq$", $\sigma(\{\mathtt{b},\mathtt{s},\mathtt{d},\mathtt{a}\}) =$ "$\top$", and $\sigma(\{\mathtt{b},\mathtt{d},\mathtt{f},\mathtt{a}\}) =$ "$\top$".

Let $\Theta' = \{x_p \sigma(R) y_I \,|\, pRI \in \Theta\}$. By Proposition A.10, it is polynomial to decide R-satisfiability of Θ'. To show that Θ is $\mathcal{V}$-satisfiable iff Θ' is R-satisfiable is analogous to the proof of Lemma A.12. □

Lemma A.14 $\mathcal{V}$-SAT($\mathcal{V}_{\neg\mathtt{a}}^{18}$) is polynomial.

Proof: By Corollary 2.4, it is sufficient to show that $\mathcal{V}$-SAT($v_{\neg\mathtt{a}}^{18}$) is polynomial. Let $r'_1 = \{\mathtt{d}\}$, $r'_2 = \{\mathtt{b},\mathtt{s},\mathtt{a}\}$, $r'_3 = \neg\mathtt{f}$, $r'_4 = \neg\mathtt{a}$ and $r'_5 = \neg\mathtt{s}$. Note that $v_{\neg\mathtt{a}}^{18} = \bigcup_{i=1}^{5}\{r'_i\}$. Furthermore, let $r_i = r'_i$ for $i \in \{1,3,4,5\}$ and let $r_2 = \{\mathtt{b},\mathtt{a}\}$. It can easily be verified that $r_k \subseteq r'_k$ for $1 \leq k \leq 5$. Furthermore, $f_{T_1}(r'_k) \subseteq r_k$ for $1 \leq k \leq 5$. Hence, by Lemma A.3, the polynomiality of $\mathcal{V}$-SAT($\mathcal{V}_{\neg\mathtt{a}}^{18}$) follows. □

Lemma A.15 $\mathcal{V}$-SAT($\mathcal{V}_{\neg\mathtt{d}}^{18}$) is polynomial.

Proof: By Corollary 2.4, it is sufficient to show that $\mathcal{V}$-SAT($v_{\neg\mathtt{d}}^{18}$) is polynomial. Let $r'_1 = \{\mathtt{d}\}$, $r'_2 = \{\mathtt{b},\mathtt{d}\}$, $r'_3 = \neg\mathtt{s}$, $r'_4 = \neg\mathtt{d}$ and $r'_5 = \neg\mathtt{f}$. Note that $v_{\neg\mathtt{d}}^{18} = \bigcup_{i=1}^{5}\{r'_i\}$. Furthermore, let $r_i = r'_i$ for $i \in \{1,3,4,5\}$ and let $r_4 = \{\mathtt{b},\mathtt{a}\}$. It can easily be verified that $r_k \subseteq r'_k$ for $1 \leq k \leq 5$. Furthermore, $f_{T_2}(r'_k) \subseteq r_k$ for $1 \leq k \leq 5$. Consequently, by Lemma A.3, the polynomiality of $\mathcal{V}$-SAT($\mathcal{V}_{\neg\mathtt{d}}^{18}$) follows. □

Lemma A.16 $\mathcal{V}$-SAT($\mathcal{V}_{\mathtt{s}}^{17}$) is polynomial.

Proof: Let Θ be an arbitrary instance of $\mathcal{V}$-SAT($\mathcal{V}_{\mathtt{s}}^{17}$). If a formula of the form $p \perp I$ is in Θ then Θ is not satisfiable. Otherwise, consider the following $\mathcal{V}$-interpretation: $\Im(p) = 0$ for every time point p and $\Im(I^-) = 0$ and $\Im(I^+) = 1$ for every interval I. Let pRI be an arbitrary formula in Θ. By the definition of $\mathcal{V}_{\mathtt{s}}^{17}$, $\mathtt{s} \in R$. Obviously, $\Im$ satisfies pRI. Since it is polynomial to check whether $pRI \in \Theta$ or not, the lemma follows. □

B INTRACTABILITY RESULTS

This section provides proofs for the NP-complete subclasses of $\mathcal{V}$ presented in Table 3. The reductions are mostly made from different subalgebras of Allen's interval algebra. Consequently, we begin this section by recapitulating some results concerning Allen's algebra.

Table 3: NP-complete subclasses of $\mathcal{V}$.

Subclass	Relations	Proof
D_1	{d}, {s}, {b, a}	Lemma B.7
D_2	{d}, {f}, {b, a}	$D_2 = \mathcal{D}_\mathcal{V}(D_1)$
D_3	{d}, {s}, {b, f, a}	Lemma B.8
D_4	{d}, {f}, {b, s, a}	$D_4 = \mathcal{D}_\mathcal{V}(D_3)$
D_5	{d}, {s}, {s, a}	Lemma B.9
D_6	{d}, {f}, {b, f}	$D_6 = \mathcal{D}_\mathcal{V}(D_5)$
D_7	{d}, {s, f}	Lemma B.10
D_8	{s}, {b, f}	Lemma B.11
D_9	{f}, {s, a}	$D_9 = \mathcal{D}_\mathcal{V}(D_8)$
D_{10}	{s}, {b, f, a}, {b, s, d, f}	$D_8 \subseteq \mathcal{C}_\mathcal{V}(D_{10})$
D_{11}	{f}, {b, s, a}, {s, d, f, a}	$D_{11} = \mathcal{D}_\mathcal{V}(D_{10})$
D_{12}	{d}, {b, f}	Lemma B.12
D_{13}	{d}, {s, a}	$D_{13} = \mathcal{D}_\mathcal{V}(D_{12})$
D_{14}	{d, a}, {b, f}	Lemma B.13
D_{15}	{b, d}, {s, a}	$D_{15} = \mathcal{D}_\mathcal{V}(D_{14})$
D_{16}	{d}, {s, f, a}	Lemma B.14
D_{17}	{d}, {b, s, f}	$D_{17} = \mathcal{D}_\mathcal{V}(D_{16})$
D_{18}	{s, f}, {b, d}	Lemma B.15
D_{19}	{s, f}, {d, a}	$D_{19} = \mathcal{D}_\mathcal{V}(D_{18})$
D_{20}	{s, f}, {b, a}	Lemma B.16
D_{21}	{d, a}, {b, s, f}	Lemma B.17
D_{22}	{b, d}, {s, f, a}	$D_{22} = \mathcal{D}_\mathcal{V}(D_{21})$
D_{23}	{s, d}, {b, f}	Lemma B.18
D_{24}	{d, f}, {s, a}	$D_{24} = \mathcal{D}_\mathcal{V}(D_{23})$
D_{25}	{s, d, a}, {b, f}	Lemma B.19
D_{26}	{b, d, f}, {s, a}	$D_{26} = \mathcal{D}_\mathcal{V}(D_{25})$
D_{27}	{b, f}, {s, a}	Lemma B.20
D_{28}	{s, f}, {b, d, a}	Lemma B.21

B.1 ALLEN'S ALGEBRA

Allen's interval algebra (Allen, 1983) is based on the notion of *relations between pairs of intervals*. An interval X is represented as an ordered pair $\langle X^-, X^+\rangle$ of real numbers with $X^- < X^+$, denoting the left and right endpoints of the interval, respectively, and relations between intervals are composed as disjunctions of *basic interval relations*. Their exact definitions can be found in (Allen, 1983). Such disjunctions are represented as sets of basic relations. The algebra is provided with the operations of *converse, intersection and composition* on intervals. The exact definitions of these operations can be found in (Allen, 1983). By the fact that there are thirteen basic relations, we get $2^{13} = 8192$ possible relations between intervals in the full algebra. We denote the set of all interval relations by $\mathcal{A}$. The reasoning problem we will consider is the problem of *satisfiability* ($\mathcal{A}$-SAT) of a set of interval variables with relations between them, *i.e.* deciding whether there exists an assignment of intervals on the real line for the interval variables, such that all of the relations between the intervals hold. Such an assignment is said to be a *$\mathcal{A}$-model* for the interval variables and relations. For $\mathcal{A}$, we have the following result.

Theorem B.1 Let $\mathcal{N}_2$ be the set $\{\{\prec, \mathtt{d}^\smile, \mathtt{o}, \mathtt{m}, \mathtt{f}^\smile\}, \{\prec, \mathtt{d}, \mathtt{o}, \mathtt{m}, \mathtt{s}\}, \{\mathtt{d}^\smile, \mathtt{o}, \mathtt{o}^\smile, \mathtt{s}^\smile, \mathtt{f}^\smile\}\}$ and let Δ_0 be

the set $\{\{\prec,\succ\},\{\equiv,\mathsf{d},\mathsf{d}^\smile,\mathsf{o},\mathsf{o}^\smile,\mathsf{m},\mathsf{m}^\smile,\mathsf{s},\mathsf{s}^\smile,\mathsf{f},\mathsf{f}^\smile\}\}$. $\mathcal{A}$-SAT($\mathcal{S}$) is NP-complete if $\mathcal{N}_2 \subseteq \mathcal{S}$ (Nebel and Bürckert, 1995), or $\Delta_0 \subseteq \mathcal{S}$ (Golumbic and Shamir, 1993).

To facilitate the forthcoming proofs we will use a closure operation for Allen's algebra which was defined in (Nebel and Bürckert, 1995).

Definition B.2 Let $S \subseteq \mathcal{A}$. Then we denote by $\mathcal{C}_\mathcal{A}(S)$ the *$\mathcal{A}$-closure* of S under converse, intersection and composition, *i.e.* the least subalgebra containing S closed under the three operations.

The key result for $\mathcal{C}_\mathcal{A}$ is the following. The proof appears in (Nebel and Bürckert, 1995).

Proposition B.3 Let $\mathcal{S} \subseteq \mathcal{A}$. Then $\mathcal{A}$-SAT($\mathcal{S}$) is polynomial iff $\mathcal{A}$-SAT($\mathcal{C}_\mathcal{A}(\mathcal{S})$) is and $\mathcal{A}$-SAT($\mathcal{S}$) is NP-complete iff $\mathcal{A}$-SAT($\mathcal{C}_\mathcal{A}(\mathcal{S})$) is.

Next, we will define a number of subclasses of $\mathcal{A}$ and prove that $\mathcal{A}$-SAT for them is NP-complete. These subclasses will be used later on in the NP-completeness proofs for the subclasses in Table 3.

Definition B.4 Let $s_0, s_1, \ldots, s_7$ be defined as follows: $s_0 = \{\mathsf{d},\mathsf{o}^\smile,\mathsf{f}\}$, $s_1 = \{\prec,\succ,\mathsf{d}^\smile,\mathsf{o},\mathsf{m},\mathsf{f}^\smile\}$, $s_2 = \{\equiv,\succ,\mathsf{s},\mathsf{s}^\smile\}$, $s_3 = \{\equiv,\mathsf{m},\mathsf{m}^\smile,\mathsf{s},\mathsf{s}^\smile,\mathsf{f},\mathsf{f}^\smile\}$, $s_4 = \{\mathsf{d},\mathsf{o},\mathsf{o}^\smile,\mathsf{s},\mathsf{f}\}$, $s_5 = \{\prec,\mathsf{d}^\smile,\mathsf{o},\mathsf{m},\mathsf{m}^\smile\}$, $s_6 = \{\equiv,\mathsf{s},\mathsf{s}^\smile\}$ and $s_7 = \{\prec,\succ,\mathsf{d}^\smile,\mathsf{o},\mathsf{o}^\smile,\mathsf{m},\mathsf{m}^\smile,\mathsf{s}^\smile,\mathsf{f}^\smile\}$. Denote by S_{ij} the set $\{s_i, s_j\}$.

Proposition B.5

Is_0J iff $J^- < I^- < J^+$;

Is_1J iff $I^- < J^- \vee I^- > J^+$;

Is_2J iff $I^- = J^- \vee I^- > J^+$;

Is_3J iff $(I^- = J^-) \vee (I^- = J^+) \vee (I^+ = J^-) \vee (I^+ = J^+)$;

Is_4J iff $(J^- < I^- < J^+) \vee (J^- < I^+ < J^+)$;

Is_5J iff $(I^- < J^-) \vee (I^- = J^+)$;

Is_6J iff $I^- = J^-$;

Is_7J iff $(I^- < J^-) \vee (I^- > J^+) \vee (I^+ < J^-) \vee (I^+ > J^+)$.

Lemma B.6 $\mathcal{A}$-SAT(S) is NP-complete for $S \in \{S_{01}, S_{02}, S_{34}, S_{56}, S_{37}\}$.

Proof: By Corollary B.3, we can study $\mathcal{A}$-SAT($\mathcal{C}_\mathcal{A}(S)$) instead of $\mathcal{A}$-SAT(S). It can be verified that $\mathcal{N}_2$ is a subset of $\mathcal{C}_\mathcal{A}(S_{01})$, $\mathcal{C}_\mathcal{A}(S_{02})$ and $\mathcal{C}_\mathcal{A}(S_{56})$. Likewise, it can be shown that Δ_0 is a subset of $\mathcal{C}_\mathcal{A}(S_{34})$ and $\mathcal{C}_\mathcal{A}(S_{37})$. Hence, NP-completeness follows from Theorem B.1. □

In the previous lemma, $\mathcal{C}_\mathcal{A}$ was computed by the utility `aclose` (Nebel and Bürckert, 1993).

B.2 NP-COMPLETE SUBCLASSES OF $\mathcal{V}$

Lemma B.7 $\mathcal{V}$-SAT(D_1) is NP-complete.

Proof: Reduction from $\mathcal{A}$-SAT(S_{01}) which is NP-complete by Lemma B.6. Let Θ be an instance of $\mathcal{A}$-SAT(S_{01}). We construct a set Θ' as follows.

1. For each formula of the type Is_0J in Θ, introduce a new time point $p_{I,J}$ and let $p_{I,J}\{\mathsf{s}\}I$ and $p_{I,J}\{\mathsf{d}\}J$ in Θ';
2. For each formula of the type Is_1J in Θ, introduce a new time point $q_{I,J}$ and let $q_{I,J}\{\mathsf{s}\}I$ and $q_{I,J}\{\mathsf{b},\mathsf{a}\}J$ in Θ'.

Clearly Θ' is an instance of the $\mathcal{V}$-SAT(D_1) problem. We show that Θ is satisfiable iff Θ'.

only-if: Assume there exists a $\mathcal{A}$-model $\Im$ of Θ. We construct an $\mathcal{V}$-interpretation $\Im'$ of Θ' as follows: $\Im'(I^-) = \Im(I^-)$, $\Im'(I^+) = \Im(I^+)$, $\Im'(p_{I,J}) = \Im(I^-)$ and $\Im'(q_{I,J}) = \Im(I^-)$.

By the construction of Θ', three types of formulae can appear in Θ'. We consider them one at a time.

1. $p_{I,J}\{\mathsf{s}\}I$ and $q_{I,J}\{\mathsf{s}\}I$. Such formulae are trivially satisfied under $\Im'$.
2. $p_{I,J}\{\mathsf{d}\}J$. If $p_{I,J}\{\mathsf{d}\}J \in \Theta'$ then $Is_0J \in \Theta$. Since $\Im$ is a $\mathcal{A}$-model of Θ, $\Im(J^+) < \Im(I^-) < \Im(J^+)$ by Proposition B.5. By the construction of $\Im'$, $\Im'(p_{I,J}) = \Im(I^-)$, $\Im'(J^-) = \Im(J^-)$ and $\Im'(J^+) = \Im(J^+)$. Hence, $p_{I,J}\{\mathsf{d}\}J$ under $\Im'$.
3. $q_{I,J}\{\mathsf{b},\mathsf{a}\}J$. If $p_{I,J}\{\mathsf{b},\mathsf{a}\}J \in \Theta'$ then $Is_1J \in \Theta$. Since $\Im$ is a $\mathcal{A}$-model of Θ, $\Im(I^-) < \Im(J^-)$ or $\Im(I^-) > \Im(J^+)$ by Proposition B.5. By the construction of $\Im'$, $\Im'(q_{I,J}) = \Im(I^-)$, $\Im'(J^-) = \Im(J^-)$ and $\Im'(J^+) = \Im(J^+)$. Hence, $q_{I,J}\{\mathsf{b},\mathsf{a}\}J$ under $\Im'$.

As a consequence, $\Im'$ is a $\mathcal{V}$-model of Θ'.

if: Assume there exists an $\mathcal{V}$-model $\Im'$ of Θ'. We construct an $\mathcal{A}$-model $\Im$ of Θ as follows: $\Im'(I^-) = \Im(I^-)$ and $\Im'(I^+) = \Im(I^+)$.

Arbitrarily choose a formula of the form Is_0J in Θ. Then there exists a time point $p_{I,J}$ in Θ' such that $p_{I,J}\{s\}I$ and $p_{I,J}\{d\}J$. It follows that $\Im'(J^-) < \Im'(I^+) < \Im'(J^+)$ and, by the construction of $\Im$, $\Im(J^-) < \Im(I^+) < \Im(J^+)$. Hence, Is_0J under $\Im$.

Arbitrarily choose a formula of the form Is_1J in Θ. Then there exists a time point $q_{I,J}$ in Θ' such that $q_{I,J}\{\mathsf{s}\}I$ and $q_{I,J}\{\mathsf{b},\mathsf{a}\}J$. It follows that $\Im'(I^-) <$

$\Im'(J^-)$ or $\Im'(I^-) > \Im'(J^+)$. By the construction of $\Im$, $\Im(I^-) < \Im(J^-)$ or $\Im(I^-) > \Im(J^+)$. Hence, Is_1J under $\Im$ by Proposition B.5.

Consequently, $\Im$ is a $\mathcal{A}$-model of Θ. We have thus shown that Θ is satisfiable iff Θ' is satisfiable. NP-completeness of $\mathcal{V}$-SAT(D_1) follows immediately. □

Lemma B.8 $\mathcal{V}$-SAT(D_3) is NP-complete.

Proof: Let $D_3 = \{\{\mathtt{d}\},\{\mathtt{s}\},\{\mathtt{b},\mathtt{f},\mathtt{a}\}\} = \{r_1', r_2', r_3'\}$. Observe that $f_{T_4}(r_1') = \{\mathtt{d}\}$, $f_{T_4}(r_2') = \{\mathtt{s}\}$ and $f_{T_4}(r_3') = \{\mathtt{b},\mathtt{a}\}$. By Lemma B.7, $D_1 = \{\{\mathtt{d}\},\{\mathtt{s}\},\{\mathtt{b},\mathtt{a}\}\}$ and $\mathcal{V}$-SAT(D_1) is NP-complete. Hence, $\mathcal{V}$-SAT(D_3) is NP-complete by Lemma A.4. □

Lemma B.9 $\mathcal{V}$-SAT(D_5) is NP-complete.

Proof sketch: Reduction from $\mathcal{A}$-SAT(S_{02}) which is NP-complete by Lemma B.6. Let Θ be an instance of $\mathcal{A}$-SAT(S_{02}). We construct a set Θ' as follows.

1. For each formula of the type Is_0J in Θ, introduce a new time point $p_{I,J}$ and let $p_{I,J}\{\mathtt{s}\}I$ and $p_{I,J}\{\mathtt{d}\}J$ in Θ';
2. For each formula of the type Is_2J in Θ, introduce a new time point $q_{I,J}$ and let let $q_{I,J}\{\mathtt{s}\}I$ and $q_{I,J}\{\mathtt{s},\mathtt{a}\}J$ in Θ'.

Clearly Θ' is an instance of the $\mathcal{V}$-SAT(D_5) problem. Show that Θ is satisfiable iff Θ' is similar to the proof of Lemma B.7. □

Lemma B.10 $\mathcal{V}$-SAT(D_7) is NP-complete.

Proof: Reduction from $\mathcal{A}$-SAT(S_{34}) which is NP-complete by Lemma B.6. Let Θ be an instance of $\mathcal{A}$-SAT(S_{34}). We construct a set Θ' as follows.

1. For each formula of the type Is_3J in Θ, introduce a new time point $p_{I,J}$ and let $p_{I,J}\{\mathtt{s},\mathtt{f}\}I$ and $p_{I,J}\{\mathtt{s},\mathtt{f}\}J$ in Θ';
2. For each formula of the type Is_3J in Θ, introduce a new time point $q_{I,J}$ and let $q_{I,J}\{\mathtt{s},\mathtt{f}\}I$ and $q_{I,J}\{\mathtt{d}\}J$ in Θ'.

Clearly Θ' is an instance of the $\mathcal{V}$-SAT(D_7) problem. We show that Θ is satisfiable iff Θ'.

Assume there exists a $\mathcal{A}$-model $\Im$ of Θ. We construct an $\mathcal{V}$-interpretation $\Im'$ of Θ' as follows:

1. $\Im'(I^-) = \Im(I^-)$;
2. $\Im'(I^+) = \Im(I^+)$;
3. $\Im'(p_{I,J}) = \Im(I^-)$ if $\Im(I^-) = \Im(J^-)$ or $\Im(I^-) = \Im(J^+)$;
4. $\Im'(p_{I,J}) = \Im(I^+)$ otherwise. In this case $\Im(I^+) = \Im(J^-)$ or $\Im(I^+) = \Im(J^+)$;
5. $\Im'(q_{I,J}) = \Im(I^-)$ if $\Im(J^-) < \Im(I^-) < \Im(J^+)$;
6. $\Im'(q_{I,J}) = \Im(I^+)$ otherwise. In this case $\Im(J^-) < \Im(I^+) < \Im(J^+)$.

By the construction of Θ', four types of formulae can appear in Θ'. We consider them one at a time.

1. $p_{I,J}\{\mathtt{s},\mathtt{f}\}I$. If $p_{I,J}\{\mathtt{s},\mathtt{f}\}I \in \Theta'$ then $Is_3J \in \Theta$. By Proposition B.5, $(I^- = J^-)$ or $(I^- = J^+)$ or $(I^+ = J^-)$ or $(I^+ = J^+)$. If $\Im(I^-) = \Im(J^-)$ or $\Im(I^-) = \Im(J^-)$ then $\Im'(p_{I,J}) = \Im(I^-)$ and, consequently, $p_{I,J}\{\mathtt{s},\mathtt{f}\}I$ under $\Im'$. Otherwise, $\Im(I^+) = \Im(J^-)$ or $\Im(I^+) = \Im(J^-)$ and $\Im'(p_{I,J}) = \Im(I^+)$. Hence $p_{I,J}\{\mathtt{s},\mathtt{f}\}I$ under $\Im'$.
2. $p_{I,J}\{\mathtt{s},\mathtt{f}\}J$. This case is analogous to the previous case.
3. $q_{I,J}\{\mathtt{s},\mathtt{f}\}I$. If $q_{I,J}\{\mathtt{s},\mathtt{f}\}I \in \Theta'$ then $Is_4J \in \Theta$. By Proposition B.5, $(\Im(J^-) < \Im(I^-) < \Im(J^+)) \vee (\Im(J^-) < \Im(I^+) < \Im(J^+))$. If $\Im(J^-) < \Im(I^-) < \Im(J^+$ then $\Im'(q_{I,J}) = \Im(I^-)$ and $q_{I,J}\{\mathtt{s}\}I$ under $\Im$. Otherwise, $\Im(J^-) < \Im(I^+) < \Im(J^+$, $\Im'(q_{I,J}) = \Im(I^+)$ and $q_{I,J}\{\mathtt{f}\}I$ under $\Im$.
4. $q_{I,J}\{\mathtt{d}\}J$. If $q_{I,J}\{\mathtt{d}\}J \in \Theta'$ then $Is_4J \in \Theta$. By Proposition B.5, $(\Im(J^-) < \Im(I^-) < \Im(J^+)) \vee (\Im(J^-) < \Im(I^+) < \Im(J^+))$. If $\Im(J^-) < \Im(I^-) < \Im(J^+$ then $\Im'(q_{I,J}) = \Im(I^-)$ and $q_{I,J}\{\mathtt{d}\}I$ under $\Im$. Otherwise, $\Im(J^-) < \Im(I^+) < \Im(J^+$, $\Im'(q_{I,J}) = \Im(I^+)$ and $q_{I,J}\{\mathtt{d}\}I$ under $\Im$.

As a consequence, $\Im'$ is a $\mathcal{V}$-model of Θ'.

Now, assume there exists a $\mathcal{V}$-model $\Im'$ of Θ'. We construct an $\mathcal{A}$-interpretation $\Im$ of Θ as follows: $\Im(I^-) = \Im'(I^-)$ and $\Im(I^+) = \Im'(I^+)$.

Arbitrarily choose a formula of the form Is_3J in Θ. Then there exists a time point $p_{I,J}$ in Θ' such that $p_{I,J}\{\mathtt{s},\mathtt{f}\}I$ and $p_{I,J}\{\mathtt{s},\mathtt{f}\}J$ under $\Im'$. This implies that one endpoint in I equals one endpoint in J and, by Proposition B.5, Is_3J under $\Im$.

Arbitrarily choose a formula of the form Is_4J in Θ. Then there exists a time point $q_{I,J}$ in Θ' such that $q_{I,J}\{\mathtt{s},\mathtt{f}\}I$ and $q_{I,J}\{\mathtt{d}\}J$ under $\Im'$. Assume $q_{I,J}\{\mathtt{s}\}I$ under $\Im'$. Then $\Im(J^-) < \Im(I^-) < \Im(J^+)$. Assume to the contrary that $p_{I,J}\{\mathtt{f}\}I$ under $\Im'$. It follows that $\Im(J^-) < \Im(I^+) < \Im(J^+)$. Hence, by Proposition B.5, Is_4J under $\Im$. □

Lemma B.11 $\mathcal{V}$-SAT(D_8) is NP-complete.

Proof sketch: Reduction from $\mathcal{A}$-SAT(S_{56}) which is NP-complete by Lemma B.6. Let Θ be an instance of $\mathcal{A}$-SAT(S_{56}). We construct a set Θ' as follows.

1. For each formula of the type Is_5J in Θ, introduce a new time point $p_{I,J}$ and let $p_{I,J}\{\mathtt{s}\}I$ and $p_{I,J}\{\mathtt{b},\mathtt{f}\}J$ in Θ';

2. For each formula of the type Is_6J in Θ, introduce a new time point $q_{I,J}$ and let $q_{I,J}\{\mathtt{s}\}I$ and $q_{I,J}\{\mathtt{s}\}J$ in Θ'.

Clearly Θ' is an instance of the $\mathcal{V}$-SAT(D_8) problem. It is a routine verification to show that Θ is satisfiable iff Θ' is satisfiable. □

Lemma B.12 $\mathcal{V}$-SAT(D_{12}) is NP-complete.

Proof: By Lemma B.9, $\mathcal{V}$-SAT(D_5) is NP-complete and $\mathcal{V}$-SAT(D_6) is NP-complete since $D_6 = \mathcal{D}_\mathcal{V}(D_5)$. Let $E = \{\{\mathtt{b}\},\{\mathtt{d}\},\{\mathtt{b},\mathtt{f}\}\}$. It can be verified that $D_6 \subseteq \mathcal{C}_\mathcal{V}(E)$ and, hence, $\mathcal{V}$-SAT(E) is NP-complete. Let Θ be an arbitrary instance of the $\mathcal{V}$-SAT(E) problem. We show how to construct an instance Θ' of the $\mathcal{V}$-SAT(D_{12}) problem that is satisfiable iff Θ is satisfiable.

We begin showing how to relate a point p_1 and an interval I_1 such as $p_1\{\mathtt{b}\}I_1$ by only using the relations in D_{12}. We introduce two fresh time points p_2 and p_3 together with two fresh time intervals I_2 and I_3. Consider the following construction: $p_1\{\mathtt{b},\mathtt{f}\}I_1$, $p_1\{\mathtt{b},\mathtt{f}\}I_2$, $p_1\{\mathtt{b},\mathtt{f}\}I_3$, $p_2\{\mathtt{b},\mathtt{f}\}I_1$, $p_2\{\mathtt{d}\}I_2$, $p_2\{\mathtt{b},\mathtt{f}\}I_3$, $p_3\{\mathtt{b},\mathtt{f}\}I_1$, $p_3\{\mathtt{b},\mathtt{f}\}I_2$ and $p_3\{\mathtt{d}\}I_3$.

We denote this set of relations with Ω. Let $\Im$ be a $\mathcal{V}$-model of Ω. For the sake of brevity we identify the time points and time intervals with their values when interpreted by $\Im$. Hence, instead of writing $\Im(p_1) < \Im(I_1^-)$, we simply write $p_1 < I_1^-$.

Obviously, $p_1 < I_1^-$ or $p_1 = I_1^+$. We begin by showing that there exists a $\mathcal{V}$-model of Ω such that $p_1 < I_1^-$. Let $\delta = (I_1^- - p_1)/5$. Consider the following assignment of values: $I_3^- = p_1 + \delta$, $p_3 = p_1 + 2\delta$, $I_2^- = p_1 + 3\delta$, $p_2 = p_1 + 4\delta$, $I_3^+ = p_1 + 4\delta$ and $I_2^+ = I_1^+$. It is not hard to see that this assignment is a $\mathcal{V}$-model of Ω.

Next, we show that there does not exist any $\mathcal{V}$-model of Ω such that $p_1 = I_1^+$. Assume $\Im$ is such a $\mathcal{V}$-model. By relation (4), we can see that $p_2 < I_1^-$ or $p_2 = I_1^+$. By assumption, $p_1 = I_1^+$. Hence, either $p_2 < I_1^-$ or $p_2 = p_1$. If $p_2 = p_1$ then relation (2) is equivalent to $p_2\{\mathtt{b},\mathtt{f}\}I_2$ which clearly contradicts relation (5). Thus, $p_2 < I_1^-$ and $p_2\{\mathtt{b}\}I_1$. By analogous reasoning one can see that $p_3 < I_1^-$ and $p_3\{\mathtt{b}\}I_1$.

Next, observe that relations (2) and (3) implies $p_1 \leq I_2^+$ and $p_1 \leq I_3^+$. Furthermore, $p_2 < I_1^-$ and $p_3 < I_1^-$ which implies $p_2 < I_1^+$ and $p_3 < I_1^-$. By our initial assumption $p_1 = I_1^+$ we get $p_2 < I_1^+ = p_1 \leq I_3^+$ and $p_3 < I_1^+ = p_1 \leq I_2^+$.

Consequently, $p_2 < I_3^+$ and $p_3 < I_2^+$. Observe that $p_2\{\mathtt{b},\mathtt{f}\}I_3$ and $p_3\{\mathtt{b},\mathtt{f}\}I_2$ by relations (6) and (8), respectively. Hence, $p_2 < I_3^-$ and $p_3 < I_2^-$.

By relations (5) and (9), $I_2^- < p_2 < I_2^+$ and $I_3^- < p_3 < I_3^+$. Hence, $p_2 < I_3^- < p_3 < I_2^- < p_2$ which is a contradiction. Consequently, every $\mathcal{V}$-model of Ω satisfies $p_1 < I_1^-$.

We have thus shown how to express the relation $\{\mathtt{b}\}$ by only using $\{\mathtt{d}\}$ and $\{\mathtt{b},\mathtt{f}\}$. Obviously, we can take an instance of the $\mathcal{V}$-SAT(E) problem and in polynomial time transform it into an equivalent instance of the $\mathcal{V}$-SAT(D_{12}) problem. NP-completeness of $\mathcal{V}$-SAT(D_{12}) follows immediately. □

Lemma B.13 $\mathcal{V}$-SAT(D_{14}) is NP-complete.

Proof sketch: By Lemma B.9, $\mathcal{V}$-SAT(D_5) is NP-complete and $\mathcal{V}$-SAT(D_6) is NP-complete since $D_6 = \mathcal{D}_\mathcal{V}(D_5)$. Let $E = \{\{\mathtt{b}\},\{\mathtt{d},\mathtt{a}\},\{\mathtt{b},\mathtt{f}\}\}$. It can be verified that $D_6 \subseteq \mathcal{C}_\mathcal{V}(E)$ and, hence, $\mathcal{V}$-SAT(E) is NP-complete. Let Θ be an arbitrary instance of the $\mathcal{V}$-SAT(E) problem. We show how to construct an instance Θ' of the $\mathcal{V}$-SAT(D_{14}) problem that is satisfiable iff Θ is satisfiable.

The proof boils down to showing how to relate a point p_1 and an interval I_1 such as $p_1\{\mathtt{b}\}I_1$ by only using the relations in D_{14}. We introduce two fresh time points p_2 and p_3 together with two fresh time intervals I_2 and I_3. Consider the following construction: $p_1\{\mathtt{b},\mathtt{f}\}I_1$, $p_1\{\mathtt{b},\mathtt{f}\}I_2$, $p_1\{\mathtt{b},\mathtt{f}\}I_3$, $p_2\{\mathtt{b},\mathtt{f}\}I_1$, $p_2\{\mathtt{d},\mathtt{a}\}I_2$, $p_2\{\mathtt{b},\mathtt{f}\}I_3$, $p_3\{\mathtt{b},\mathtt{f}\}I_1$, $p_3\{\mathtt{b},\mathtt{f}\}I_2$ and $p_3\{\mathtt{d},\mathtt{a}\}I_3$. It is fairly straightforward to show that p_1 can only be related to I_1 with the relation $\mathtt{b}$. Hence, we can take an instance of the $\mathcal{V}$-SAT(E) problem and in polynomial time transform it into an equivalent instance of the $\mathcal{V}$-SAT(D_{14}) problem. NP-completeness of $\mathcal{V}$-SAT(D_{14}) follows immediately. □

Lemma B.14 $\mathcal{V}$-SAT(D_{16}) is NP-complete.

Proof: Reduction from $\mathcal{V}$-SAT(D_{13}). Use the model transformation T_4 and apply Lemma A.4. □

Lemma B.15 $\mathcal{V}$-SAT(D_{18}) is NP-complete.

Proof sketch: By Lemma B.10, $\mathcal{V}$-SAT(D_7) is NP-complete. Let $E = \{\{\mathtt{s}\},\{\mathtt{s},\mathtt{f}\},\{\mathtt{b},\mathtt{d}\}\}$. It can be verified that $D_7 \subseteq \mathcal{C}_\mathcal{V}(E)$ and, hence, $\mathcal{V}$-SAT(E) is NP-complete. Let Θ be an arbitrary instance of the $\mathcal{V}$-SAT(E) problem.

We show how to relate a time point p_1 and an interval I_1 with the relation $\mathtt{s}$. Introduce two fresh time points p_2 and p_3 together with a fresh time interval I_2. Consider the following construction: $p_1\{\mathtt{s},\mathtt{f}\}I_1$, $p_1\{\mathtt{b},\mathtt{d}\}I_2$, $p_2\{\mathtt{b},\mathtt{d}\}I_1$, $p_2\{\mathtt{s},\mathtt{f}\}I_2$, $p_3\{\mathtt{s},\mathtt{f}\}I_1$ and $p_3\{\mathtt{s},\mathtt{f}\}I_2$.

It is not hard to show that p_1 can only be related to I_1 with the relation $\mathtt{s}$. Consequently, we can take an instance of the $\mathcal{V}$-SAT(E) problem and in polynomial time transform it into an equivalent instance of the $\mathcal{V}$-SAT(D_{18}) problem. NP-completeness of $\mathcal{V}$-SAT(D_{18}) follows immediately. □

Lemma B.16 $\mathcal{V}$-SAT(D_{20}) is NP-complete.

Proof sketch: Reduction from $\mathcal{A}$-SAT(S_{37}) which is NP-complete by Lemma B.6. Let Θ be an instance of $\mathcal{A}$-SAT(S_{37}). We construct a set Θ' as follows.

1. For each formula of the type Is_3J in Θ, introduce a new time point $p_{I,J}$ and let $p_{I,J}\{\mathtt{s},\mathtt{f}\}I$ and $p_{I,J}\{\mathtt{s},\mathtt{f}\}J$ in Θ';
2. For each formula of the type Is_3J in Θ, introduce a new time point $q_{I,J}$ and let $q_{I,J}\{\mathtt{s},\mathtt{f}\}I$ and $q_{I,J}\{\mathtt{b},\mathtt{a}\}J$ in Θ'.

Clearly Θ' is an instance of the $\mathcal{V}$-SAT(D_{20}) problem. Proving that Θ is satisfiable iff Θ' is similar to the proof of Lemma B.10. □

Lemma B.17 $\mathcal{V}$-SAT(D_{21}) is NP-complete.

Proof: Reduction from $\mathcal{V}$-SAT(D_{14}). Use the model transformation T_1 and apply Lemma A.4. □

Lemma B.18 $\mathcal{V}$-SAT(D_{23}) is NP-complete.

Proof: Reduction from $\mathcal{V}$-SAT(D_{12}). Use the model transformation T_3 and apply Lemma A.4. □

Lemma B.19 $\mathcal{V}$-SAT(D_{25}) is NP-complete.

Proof: Reduction from $\mathcal{V}$-SAT(D_{14}). Use the model transformation T_3 and apply Lemma A.4. □

Lemma B.20 $\mathcal{V}$-SAT(D_{27}) is NP-complete.

Proof sketch: By Lemma B.10, $\mathcal{V}$-SAT(D_7) is NP-complete. Let $E = \{\{\mathtt{b}\},\{\mathtt{b},\mathtt{f}\},\{\mathtt{s},\mathtt{a}\}\}$. It can be verified that $D_5 \subseteq \mathcal{C}_\mathcal{V}(E)$ and, hence, $\mathcal{V}$-SAT(E) is NP-complete. Let Θ be an arbitrary instance of $\mathcal{V}$-SAT(E). We show how to construct an instance Θ' of $\mathcal{V}$-SAT(D_{27}) that is satisfiable iff Θ is satisfiable.

We begin by showing how to relate a point p_1 and an interval I_1 such as $p_1\{\mathtt{b}\}I_1$ by only using the relations in D_{27}. We introduce two fresh time points p_2 and p_3 together with a fresh time interval I_2. Consider the following construction: $p_1\{\mathtt{b},\mathtt{f}\}I_1$, $p_1\{\mathtt{b},\mathtt{f}\}I_2$, $p_2\{\mathtt{b},\mathtt{f}\}I_1$, $p_2\{\mathtt{s},\mathtt{a}\}I_2$, $p_3\{\mathtt{s},\mathtt{a}\}I_1$ and $p_3\{\mathtt{b},\mathtt{f}\}I_2$.

It is fairly straightforward to show that p_1 can only be related to I_1 with the relation b. Hence, we can take an instance of the $\mathcal{V}$-SAT(E) problem and in polynomial time transform it into an equivalent instance of the $\mathcal{V}$-SAT(D_{27}) problem. NP-completeness of $\mathcal{V}$-SAT(D_{27}) follows immediately. □

Lemma B.21 $\mathcal{V}$-SAT(D_{28}) is NP-complete.

Proof: Reduction from GRAPH 3-COLOURABILITY, which is NP-complete. Let $G = \langle V, E\rangle$ be an arbitrary undirected graph.

In the proof we will make repeated use of the concept of a *separator*, a construction which forces two points to have distinct values in all models. Given two points p, q we construct a separator by introducing a new interval I and adding the relations $p\{\mathtt{s},\mathtt{f}\}I$ and $q\{\mathtt{b},\mathtt{d},\mathtt{a}\}I$. Clearly, all models $\mathfrak{S}$ must satisfy $\mathfrak{S}(p) \neq \mathfrak{S}(q)$.

We now construct the set of relations stepwise. First, we construct a *paint-box* by introducing two points p_1 and p_2, two intervals I_1 and I_2 plus the relations

$$p_1\{\mathtt{s},\mathtt{f}\}I_1,\ p_1\{\mathtt{s},\mathtt{f}\}I_2,\ p_2\{\mathtt{s},\mathtt{f}\}I_1,\ p_2\{\mathtt{b},\mathtt{d},\mathtt{a}\}I_2$$

over these. Note that the interval I_2 acts as a separator for p_1 and p_2, which are thus forced to take on different values. Further, the intervals I_1 and I_2 must have some common end-point, coinciding with p_1. We use the constant r to denote this value. Hence, the remaining end-point of I_1 must coincide with p_2 and the remaining end-point of I_2 must be distinct from both p_1 and p_2. We denote the values of these two remaining end-points g and b respectively. We can think of the values r, g and b as colours, constituting our palette. Of course, the actual denotations of these three values differ between models, but the important thing is only that they denote three distinct values in each and every model.

Now, for each vertex $v_i \in V$, we construct a *selector* consisting of three points q_i^0, q_i^1 and q_i^2 plus two intervals J_i^0 and $J_i^{1,2}$, connected as follows. First introduce a separator for q_i^1 and q_i^2, using interval $J_i^{1,2}$, *i.e.* introduce the relations $q_i^1\{\mathtt{s},\mathtt{f}\}J_i^{1,2}$, $q_i^2\{\mathtt{b},\mathtt{d},\mathtt{a}\}J_i^{1,2}$. Then connect the points to the remaining interval by adding the relations $q_i^0\{\mathtt{s},\mathtt{f}\}J_i^0$, $q_i^1\{\mathtt{s},\mathtt{f}\}J_i^0$, $q_i^2\{\mathtt{s},\mathtt{f}\}J_i^0$. Finally, connect this whole gadget to the paint-box by adding the relations $q_i^1\{\mathtt{s},\mathtt{f}\}I_1$, $q_i^2\{\mathtt{s},\mathtt{f}\}I_2$. The selector works as follows. The endpoints of I_1 correspond to the colours r and g, so q_i^1 is forced to have either of these values. Similarly, q_i^2 must have either of the values r and b. Now, q_i^1 and q_i^2 are separated, so together they select a subpalette of two colours, assigning one colour each to the end-points of J_i^0. Finally, q_i^0 selects one of these two colours. So far, there are no further constraints, so q_i^0 may be freely assigned any of the three colours from our palette.

Finally, for each edge $\{v_i, v_j\} \in E$ we introduce a separator, consisting of the new interval $K_{i,j}$ and the two relations $q_i^0\{\mathtt{s},\mathtt{f}\}K_{i,j}$, $q_j^0\{\mathtt{b},\mathtt{d},\mathtt{a}\}K_{i,j}$, preventing q_i^0 and q_j^0 to have the same value whenever there is an edge between the vertices v_i and v_j.

It is obvious that G is 3-colourable iff the network just constructed is satisfiable, so NP-completeness follows. □

Comparing Space Efficiency of Propositional Knowledge Representation Formalisms

Marco Cadoli, Francesco M. Donini, Paolo Liberatore, Marco Schaerf
Dipartimento di Informatica e Sistemistica,
Università di Roma "La Sapienza",
Via Salaria 113, I-00198, Roma, Italy
{cadoli|donini|liberatore|schaerf}@dis.uniroma1.it

Abstract

We investigate the *space efficiency* of a Propositional Knowledge Representation (PKR) formalism. Informally, the space efficiency of a formalism F in representing a certain piece of knowledge α, is the size of the shortest formula of F that represents α. In this paper we assume that knowledge is either a set of propositional interpretations or a set of formulae (theorems). We provide a formal way of talking about the relative ability of PKR formalisms to compactly represent a set of models or a set of theorems. We introduce two new compactness measures, the corresponding classes, and show that the relative space efficiency of a PKR formalism in representing models/theorems is directly related to such classes. In particular, we consider formalisms for nonmonotonic reasoning, such as circumscription and default logic, as well as belief revision operators.

1 INTRODUCTION

Motivations. During the last years a large number of formalisms for knowledge representation (KR) have been proposed in the literature. Given some (informal) knowledge of a domain, a knowledge engineer has to choose the most appropriate KR formalism to represent it. Formalisms can be chosen for their semantics, the complexity of inference, or other properties. Here we investigate their *space efficiency*. Informally, the space efficiency of a formalism F in representing a certain piece of knowledge α, is the size of the shortest formula of F that represents α. Space efficiency –also called *succinctness* or *compactness*– of a formalism can be measured wrt its ability to represent various forms of knowledge. In this paper we focus on propositional KR (PKR) formalisms, and assume that knowledge is either a set of propositional interpretations or a set of formulae (theorems). Consequently, we consider a PKR reasoning problem to be either model checking, or theorem proving.

Conceptually similar investigations are made in the field of relational database query languages. In that field one of the goals is to know exactly what information it is possible to extract from a database by means of a query. The typical result of such investigations characterizes the *expressiveness* of a query language, often saying that language A captures more/less/same queries than/as language B. Sometimes such results are conditional to non-collapse of some complexity classes.

State of the Art. Most PKR formalisms are "translatable" one into another, although such translation may lead to an exponential increase of the size of the formula. Most of the translations have Propositional Logic (PL) as their target formalism. E.g., [BED91] from default logic to PL, [BED94] from disjunctive logic programs to PL, [Win89] from revised knowledge bases to PL, [GPP89] from circumscription to PL.

Only very recently researchers started analyzing the space efficiency of PKR formalisms; this kind of investigation includes questions such as "is exponential increase of the above mentioned translations intrinsic, or is it possible to design a polynomial-size translation?" In [CDS95] it was shown that many interesting fragments of default logic and circumscription cannot be expressed by polynomial-time fragments of PL without super-polynomially increasing the size of formulae. It was proven that super-polynomial increase of the size is necessary when translating unrestricted propositional circumscription [CDSS95] and most operators for belief revision into PL [CDLS95, Lib95]. In [GKPS95] Gogic, Kautz, Papadimitriou and Selman analyzed the relative succinctness of several PKR for-

malisms in representing sets of models. Among other results, they showed that skeptical default logic represents sets of models more succinctly than circumscription. Unfortunately, all the above results are based on ad-hoc proofs and do not help us to define equivalence classes for the space efficiency of KR formalisms.

In a recent paper [CDLS96], we have introduced a new complexity measure for decision problems, called *compilability*. In the present paper we show how this new measure can be directly used to characterize the space efficiency of PKR formalisms.

Goal. In KR the notion of *polynomial-time solvability* models the concept of *tractable* reasoning problem. Analogously, the notion of *polynomial many-one reducibility* models the relation existing between two reasoning problems whose time complexity is comparable. The latter notion allows one to say, e.g., that inference in PL is one of the hardest problems among those in coNP. Our goal is to provide a formal way of talking about the relative ability of PKR formalisms to compactly represent information, where the information is either a set of models or a set of theorems. In particular, we would like to be able to say that a specific PKR formalism provides "one of the most compact ways to represent models/theorems" among the PKR formalisms of a specific class.

Results. We introduce two new compactness measures (*model* and *theorem compactness*) and the corresponding classes (model-C and thm-C, where C is a complexity class like P, NP, coNP, etc.). Such classes form two hierarchies that are isomorphic to the polynomial-time hierarchy [Sto76]. We show that the relative space efficiency of a PKR formalism is directly related to such classes. In particular, the ability of a PKR formalism to compactly represent sets of models/theorems is directly related to which classes of the model/theorem hierarchy it belongs to. Problems higher up in the model/theorem hierarchy can represent sets of models/theorems more compactly than formalisms that are in lower classes.

This classification is obtained through a general framework, and not by making direct comparisons, and ad-hoc proofs, between the various PKR formalisms. Furthermore, our approach also allows for a simple and intuitive notion of completeness for both model and theorem hierarchies. This notion precisely characterizes both the relation between formalisms at different levels, and the relations between problems at the same level. An interesting result is that two PKR formalisms in which model checking or inference belong to the same time complexity may belong to different compactness classes. This may suggest a criterion for choosing between two PKR formalisms in which reasoning has the same time complexity—namely, choose the more compact one. Also, two PKR formalisms may belong to the same theorem compactness class, yet to different model compactness classes. This stresses the importance of first clarifying whether one wants to represent models or theorems when choosing a PKR formalism.

Outline. In the next section we briefly recall some notions on non-uniform computation that are important for what follows. In Section 3 we give basic definitions and assumptions about PKR formalisms, and we propose two general definitions about translations between formalisms. In Section 4 we recall compilability classes from [CDLS96]. In Section 5 we show how compilability classes can be used to compare the space efficiency of PKR formalisms, and in Section 6 we actually compare many known PKR formalisms using our framework. Finally, some conclusions are drawn.

2 NON-UNIFORM COMPUTATION

We assume the reader is familiar with basic complexity classes, such as P, NP and (uniform) classes of the polynomial hierarchy (see, for example, [GJ79]). Here we just briefly introduce non-uniform classes, following Johnson [Joh90].

Definition 1 *An* advice-taking Turing machine *is a Turing machine that has associated with it a special "advice oracle" A, which can be any function (not necessarily a recursive one). On input s, a special "advice tape" is automatically loaded with $A(|s|)$ and from then on the computation proceeds as normal, based on the two inputs, x and $A(|s|)$.*

Note that the advice is only function of the *size* of the input, not of the input itself.

Definition 2 *An advice-taking Turing machine uses* polynomial advice *if its advice oracle A satisfies $|A(n)| \leq p(n)$ for some fixed polynomial p and all non-negative integers n.*

Definition 3 *If* C *is a class of languages defined in terms of resource-bounded Turing machines, then* C/poly *is the class of languages defined by Turing machines with the same resource bounds but augmented by polynomial advice.*

Any class C/poly is also known as *non-uniform* C, where non-uniformity is due to the presence of the advice. Non-uniform and uniform complexity classes are related in [KL80, Yap83]. In particular, Karp and

Lipton proved in [KL80] that if NP $\subseteq$ P/poly then $\Pi_2^p = \Sigma_2^p$ = PH, i.e., the polynomial hierarchy collapses at the second level, while Yap in [Yap83, pg. 292 and Theorem 2] generalized their results showing that if NP $\subseteq$ coNP/poly then $\Pi_3^p = \Sigma_3^p$ = PH, i.e., the polynomial hierarchy collapses at the third level. Such a collapse is considered very unlikely by most researchers in structural complexity.

3 PROPOSITIONAL KR FORMALISMS

We consider a finite alphabet of propositional symbols $L = \{a, b, c, \ldots\}$, possibly with subscripts. We admit connectives of fixed arity for constructing well-formed formulae, as an example, $\wedge$ is the standard binary conjunction, the symbol for defaults $\frac{:}{}$ of default logic [Rei80] is a ternary connective, etc. Since we restrict to propositional formalisms, we admit neither variable symbols, nor quantifiers. We distinguish between two kinds of formulae: *knowledge bases* and *queries*. The languages describing well-formed knowledge bases and well-formed queries may be different within the same formalism, e.g., this is the case for default logic, or logic programming. An *interpretation* for L is a mapping from L in {true, false}. A *model-theoretic semantics* for a formalism is a description of how to extend an interpretation for L to well-formed knowledge bases and queries. Observe that the semantics of a formula needs not to be obtained in an easy way from the semantics of its components and connectives: E.g., for skeptical default logic an interpretation I assigns true to a default theory $\langle D, W\rangle$ iff there is a (propositional) extension of $\langle D, W\rangle$ such that I assigns true to all of its formulae. A *model* of a knowledge base KB in a formalism F is an interpretation M that maps KB to true (written $M \models_F KB$), and similarly for a query Q (written $M \models_F Q$). Sometimes models will be denoted as sets of letters which are mapped into true. A *proof theory* of a formalism F is a definition of which queries are derivable from which knowledge bases in F. When a query Q is derivable from a knowledge base KB in F, we call Q a *theorem* of KB (written $KB \vdash_F Q$). Observe that some formalisms have only a proof theory, with no model-theoretic semantics, e.g., credulous default logic. When a formalism F has both a model-theoretic semantics and a proof theory, we impose the usual relation between them: $KB \vdash_F Q$ iff $\forall M : M \models_F KB$ implies $M \models_F Q$. When F is classical propositional logic PL, we omit the subscript from $\vdash$ and $\models$.

Assumption 1. Throughout the paper, we assume that *a knowledge base in a PKR formalism is used to represent either its set of models, or its set of theorems, or both.*

In this way, all PKR formalisms can be compared on the basis of which sets of models or theorems they can represent, and how succinct is the representation.

Assumption 2. As for queries, *we consider only queries whose size is less than or equal to that of the knowledge base.*

As a consequence, propositional tautologies whose size is larger than that of the knowledge base are not considered among the theorems.

3.1 REDUCTIONS AMONG KR FORMALISMS

We now define the forms of reduction between PKR formalisms that we analyze in the following sections. A formula can always be represented as a string over an alphabet Σ, hence from now on we consider translations as functions transforming strings. We assume that Σ contains a special symbol # to denote blanks. The *length* of a string $x \in \Sigma^*$ is denoted by $|x|$.

A function f is called *poly-size* if there exists a polynomial p such that for all x it holds $|f(x)| \leq p(|x|)$. Moreover, a function g is called *poly-time* if there exists a polynomial q such that for all x, $g(x)$ can be computed in time less than or equal to $q(|x|)$. These definitions extend to binary functions considering the sum of the sizes of their arguments.

Let F_1 and F_2 be two PKR formalisms. There exists a *poly-size reduction* from F_1 to F_2, denoted as $f : F_1 \mapsto F_2$, if f is a poly-size function such that for any given knowledge base KB in F_1, $f(KB)$ is a knowledge base in F_2.

Clearly some restrictions must be imposed on such functions. In particular, we define a form of reduction that preserves the models of the original theory:

Definition 4 (Model Preservation) *A poly-size reduction $f : F_1 \mapsto F_2$ satisfies* model-preservation *if for each knowledge base KB in F_1 there exists a poly-time function g_{KB} such that for every interpretation M of the variables of KB it holds that $M \models_{F_1} KB$ iff $g_{KB}(M) \models_{F_2} f(KB)$.*

Example We reduce a fragment of skeptical default logic to circumscription with varying letters, using the transformation shown in [Eth87]. Let $\langle D, W\rangle$ be a *prerequisite-free normal* (PFN) default theory, i.e., all defaults are of the form $\frac{:\gamma}{\gamma}$, where γ is a generic formula. Let Z be the set of letters occurring in $\langle D, W\rangle$. Define P_D as the set of letters $\{a_\gamma | \frac{:\gamma}{\gamma} \in D\}$.

The function f can be defined in the following way: $f(\langle D, W\rangle) = CIRC(T; P_D; Z)$, where $T = W \cup \{a_\gamma \equiv \neg\gamma | a_\gamma \in P_D\}$, P_D are the letters to be minimized, and Z (the set of letters occurring in $\langle D, W\rangle$) are varying letters. We show that f is a model-preserving poly-size reduction. In fact, given a set of PFN defaults D let g_D be a function such that for each interpretation M for Z, $g_D(M) = M \cup \{a_\gamma \in P_D | M \models \neg\gamma\}$. Clearly, f is poly-size, g_D is poly-time, and M is a model of at least one extension of $\langle D, W\rangle$ iff $g_D(M) \models CIRC(T; P_D; Z)$. The subscript D of g_D stresses the fact that the function g_D does not depend in this case on the whole knowledge base $\langle D, W\rangle$, but just on D.

A weaker form of reduction is the following one, where only theorems are preserved:

Definition 5 (Theorem Preservation) *A poly-size reduction $f : F_1 \mapsto F_2$ satisfies* theorem-preservation *if for each knowledge base KB in F_1 there exists a poly-time function g_{KB} such that for every query Q on the variables of KB it holds that $KB \vdash_{F_1} Q$ iff $f(KB) \vdash_{F_2} g_{KB}(Q)$.*

An example of theorem-preserving (and non-model-preserving) poly-size reduction from updated knowledge bases to PL is given in [Win89]. The reduction shown in the previous example is also theorem-preserving, using for g_{KB} the identity function.

We remark that our definitions of reduction are more general than those proposed in [GKPS95]. In particular, in [GKPS95] only a notion analogous to Definition 4 is considered, and only for the case when g_{KB} is the identity – i.e., models in the two formalisms should be identical.

4 COMPILABILITY CLASSES

In this section we summarize some definitions and results proposed in [CDLS96] adapting them to the context and terminology of PKR formalisms. In that paper we introduced a new complexity measure for decision problems, called *compilability*. Following the intuition that a knowledge base is known well before questions are posed to it, we divide a reasoning problem into two parts: one part is *fixed* or *accessible off-line* (the knowledge base), and the second one is *variable*, or *accessible on-line* (the model/query). Compilability aims at capturing the *on-line complexity* of solving a problem composed of such inputs, i.e., complexity wrt the second input when the first one can be preprocessed in an arbitrary way. In the next section we show the close connection between compilability and the space efficiency of PKR formalisms.

We define a *language of pairs* S as a subset of $\Sigma^* \times \Sigma^*$. This is necessary to represent the two inputs to a PKR reasoning problem, i.e., the knowledge base (KB), and the query or interpretation. As an example, the problem CNF CLAUSE INFERENCE (CI) is defined as

$$\text{CI} = \{\langle x, y\rangle \mid x \text{ is a CNF propositional formula, } y \text{ is a clause and } x \vdash y\}$$

It is well known that CI is coNP-complete wrt the sum of the size of both inputs, i.e., it is one of the "hardest" problems among those belonging to coNP. Our goal is to prove that CI is the "hardest" theorem-proving problem among those that can be solved preprocessing in an arbitrary way the first input, i.e., the KB. To this end, we introduce a new hierarchy of classes, the *non-uniform compilability classes*, denoted as nu-comp-C, where C is a generic uniform complexity class, such as NP, coNP, Σ_2^p, ...

Definition 6 (nu-comp-C Classes) *A language of pairs $S \subseteq \Sigma^* \times \Sigma^*$ belongs to* nu-comp-C *iff there exists a binary poly-size function f and a language of pairs S' such that for all $\langle x, y\rangle \in S$ it holds:*

1. $\langle f(x, |y|), y\rangle \in S'$ *iff* $\langle x, y\rangle \in S$;
2. $S' \in$ C.

Notice that the poly-size function f takes as input both x (the KB) and the size of y (the query or interpretation). If we want to rewrite off-line x into a new string (formula), our definition requires that we know in advance the *size* of y. If y is an interpretation of the letters of x, its size is bounded by $|x|$ and therefore it is known in advance. However, if y is a query it is not very reasonable to assume that, at compilation time, we know the size of y. Anyway, the only information necessary at compilation time is an *upper bound* of the size of all y's that can occur. For this reason, we assume that the size of each query is less than or equal to the size of the knowledge base (cf. Assumption 2).

For each C, the class nu-comp-C generalizes the non-uniform class C/poly —i.e., C/poly $\subset$ nu-comp-C — by allowing for a fixed part x. In Figures 1 and 2 we compare the machines corresponding to C/poly and nu-comp-C.

We introduce now a reduction between problems.

Definition 7 (Non-uniform comp-reducibility) *Given two problems A and B, A is* non-uniformly comp-reducible *to B (denoted as $A \leq_{nu-comp} B$) iff there exist two poly-size binary functions f_1 and f_2,*

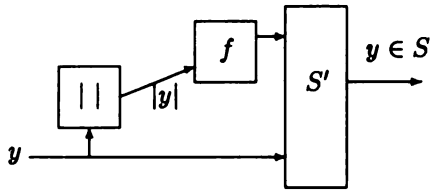


Figure 1: The C/poly Machine

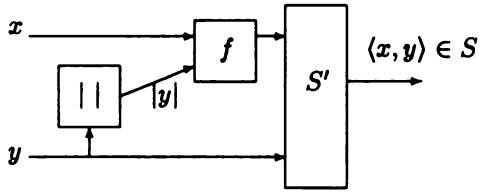
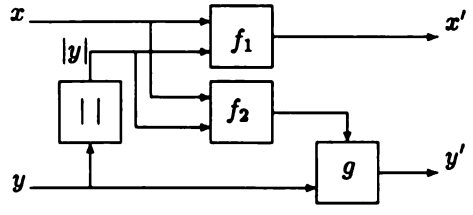


Figure 2: The nu-comp-C Machine

and a polynomial-time binary function g such that for every pair $\langle x, y\rangle$ it holds that $\langle x, y\rangle \in A$ if and only if $\langle f_1(x, |y|), g(f_2(x, |y|), y)\rangle \in B$.

The $\leq_{nu-comp}$ reductions can be pictorially represented as follows:



Such reductions satisfy all important properties of a reduction:

Theorem 1 *The reductions $\leq_{nu-comp}$ satisfy transitivity and are compatible (in the sense of Johnson [Joh90, pg. 79]) with the class* nu-comp-C *for every complexity class* C.

Therefore, it is possible to define the notions of *hardness* and *completeness* for nu-comp-C for every complexity class C.

Definition 8 (nu-comp-C-completeness) *Let S be a language of pairs and* C *a complexity class. S is* nu-comp-C-hard *iff for all problems $A \in$* nu-comp-C *we have that $A \leq_{nu-comp} S$. Moreover, S is* nu-comp-C-complete *if S is in* nu-comp-C *and is* nu-comp-C-hard.

We now have the right complexity class to completely characterize the problem CI. In fact CI is nu-comp-coNP-complete. Furthermore, the hierarchy formed by the compilability classes is proper if and only if the polynomial hierarchy is proper [CDLS96, KL80, Yap83] (which is widely conjectured to be true).

We close the section by giving a rationale for the complexity classes we defined. Informally we may say that nu-comp-NP-hard problems are "not compilable to P", as from the above considerations we know that if there exists a preprocessing of their fixed part that makes them on-line solvable in polynomial time, then the polynomial hierarchy collapses. The same holds for nu-comp-coNP-hard problems. In general, a problem which is nu-comp-C-complete for a class C containing P can be regarded as the "toughest" problem in C, even after arbitrary preprocessing of the fixed part. On the other hand, a problem in nu-comp-C is a problem that, after preprocessing of the fixed part, becomes a problem in C (i.e., it is "compilable to C").

5 SUCCINCTNESS OF PKR FORMALISMS

In this section we show how to use our compilability classes to compare the succinctness of PKR formalisms.

Let F_1 and F_2 be two formalisms representing sets of models. We prove in this section that any knowledge base α in F_1 can be reduced, via a poly-size reduction, to a knowledge base β in F_2 satisfying model-preservation if and only if the compilability class of the problem of *model checking* (first input: KB, second input: interpretation) in F_1 is higher than or equal to the compilability class of the problem of model checking in F_2.

Similarly, we prove that theorem-preserving poly-size reductions exist if and only if the compilability class of the problem of *inference* (first input: KB, second input: query, cf. definition of CI) in F_1 is higher than or equal to the compilability class of the problem of inference in F_2.

In order to simplify the presentation and proof of the theorems we introduce some definitions.

Definition 9 (Model/Theorem hardness/completeness) *Let F be a PKR formalism. If the problem of model checking for F belongs to the compilability class* nu-comp-C, *we say that F is in* model-C. *Similarly, if model checking is* nu-comp-C-*complete (hard), we say that F is* model-C-*complete (hard). If the problem of inference for the formalism F belongs to the compilability class* nu-comp-C, *we say that F is in* thm-C. *Similarly, if inference is* nu-comp-C-*complete (hard), we say that F is* thm-C-*complete (hard).*

These definitions implicitly define two hierarchies, the model hierarchy (model-C) and the theorem hierarchy (thm-C). As an example rephrased from [CDS95, Thm. 2], we characterize model and theorem classes of propositional logic (PL).

Theorem 2 PL *is in* model-P *and* thm-coNP-*complete.*

We can now formally establish the connection between succinctness of representations and compilability classes. In the following theorems, the complexity classes C, C_1, C_2 belong to the polynomial hierarchy [Sto76]. In Theorems 4 and 6 we assume that the polynomial hierarchy does not collapse.

Theorem 3 *Let F_1 and F_2 be two PKR formalisms. If F_1 is* model-C-*complete and F_2 is* model-C-*hard, then there exists a poly-size reduction $f : F_1 \mapsto F_2$ satisfying model preservation.*

Proof. (sketch) Recall that since F_1 is model-C-complete, model checking in F_1 is in nu-comp-C, and since F_2 is model-C-hard, model checking in F_1 is non-uniformly comp-reducible to model checking in F_2. That is, (adapting Def. 7) there exist two poly-size binary functions f_1 and f_2, and a polynomial-time binary function g such that for every pair $\langle KB, M\rangle$ it holds that $M \models_{F_1} KB$ if and only if $g(f_2(KB, |M|), M) \models_{F_2} f_1(KB, |M|)$. Now observe that $|M|$ can be computed from KB by simply counting the letters appearing in KB; let f_3 be such a counting function, i.e., $|M| = f_3(KB)$. Clearly, f_3 is poly-size. Define the reduction f as $f(KB) = f_1(KB, f_3(KB))$. Since poly-size functions are closed under composition, f is poly-size. We show that f is a model-preserving reduction. In fact, given KB, one can compute $z = f_2(KB, |M|) = f_2(KB, f_3(KB))$. Since f_2 and f_3 are poly-size, z has polynomial size wrt $|KB|$, hence wrt $|M|$. Define $h_{KB}(M) = g(z, M)$. Clearly, h_{KB} is a poly-time function, and from its construction, $M \models_{F_1} KB$ iff $h_{KB}(M) \models_{F_2} f(KB)$. □

Theorem 4 *Let F_1 and F_2 be two PKR formalisms. If the polynomial hierarchy does not collapse, F_1 is* model-C_1-*hard, F_2 is in* model-C_2, *and $C_2 \subset C_1$, then there is no poly-size reduction $f : F_1 \mapsto F_2$ satisfying model preservation.*

Proof. (sketch) We show that if such a reduction exists, then C_1/poly $\subseteq$ C_2/poly and the polynomial hierarchy collapses at some level (see [Yap83]). Let A be a complete problem for class C_1, e.g., if C_1 is Σ_3^p then A may be validity of $\exists\forall\exists$-quantified boolean formulae [Sto76]. Define the problem ϵA as:

$$\epsilon A = \{\langle x, y\rangle \mid x = \epsilon \text{ (the empty string) and } y \in A\}$$

From [CDLS96, Thm. 6], ϵA is nu-comp-C_1-complete. Since model checking in F_1 is model-C_1-hard, ϵA is non-uniformly comp-reducible to model checking in F_1. That is, (adapting Def. 7) there exist two poly-size binary functions f_1 and f_2, and a polynomial-time binary function g such that for every pair $\langle\epsilon, y\rangle$, $\langle\epsilon, y\rangle \in \epsilon A$ if and only if $g(f_2(\epsilon, |y|), y) \models_{F_1} f_1(\epsilon, |y|)$. Let $|y| = n$. Clearly, the knowledge base $f_1(\epsilon, |y|)$ depends just on n, i.e., there is one knowledge base for each integer. Call it KB_n. Moreover, also $f_2(\epsilon, |y|) = f_2(\epsilon, n)$ depends just on n: call it O_n (for Oracle). Observe that O_n has polynomial size wrt n.

If there exists a poly-size reduction $f : F_1 \mapsto F_2$ satisfying model preservation, then given the knowledge base KB_n there exists a poly-time function h_n such that $g(O_n, y) \models_{F_1} KB_n$ if and only if $h_n(g(O_n, y)) \models_{F_2} f(KB_n)$.

Therefore, the nu-comp-C_1-complete problem ϵA can be non-uniformly reduced to a problem in nu-comp-C_2 as follows: Given y, from its size $|y| = n$ one obtains (with an arbitrary preprocessing) $f(KB_n)$ and O_n. Then one checks whether the interpretation $h_n(g(O_n, y))$ (computable in poly-time given y and O_n) is a model in F_2 for $f(KB_n)$. From the fact that model checking in F_2 is in nu-comp-C_2, we have that nu-comp-$C_1 \subseteq$ nu-comp-C_2. From [CDLS96, Thm. 9] it follows that C_1/poly $\subseteq$ C_2/poly, and from [Yap83] the polynomial hierarchy collapses. □

The above theorems show that the hierarchy of classes model-C exactly characterizes the space efficiency of a formalism in representing sets of models. In fact, two formalisms at the same level in the model hierarchy can be reduced one into the other via a poly-size reduction (Theorem 3), while there is no poly-size reduction from a formalism (F_1) higher up in the hierarchy into one (F_2) in a lower class (Theorem 4). In the latter case we say that F_1 is *more space-efficient* than F_2.

Analogous results (with similar proofs) hold for poly-size reductions preserving theorems.

Theorem 5 *Let F_1 and F_2 be two PKR formalisms. If F_1 is* thm-C-*complete and F_2 is* thm-C-*hard, then there exists a poly-size reduction $f : F_1 \mapsto F_2$ satisfying theorem preservation.*

Theorem 6 *Let F_1 and F_2 be two PKR formalisms. If the polynomial hierarchy does not collapse, F_1 is* thm-C_1-*hard, F_2 is in* thm-C_2, *and $C_2 \subset C_1$, then*

there is no poly-size reduction $f : F_1 \mapsto F_2$ satisfying theorem preservation.

Theorems 3-6 show that compilability classes characterize very precisely the relative capability of PKR formalisms to represent sets of models or sets of theorems. For example, as a consequence of Theorems 2 and 6 there is no poly-size reduction from PL to Horn clauses that preserves the theorems unless the polynomial hierarchy collapses. Kautz and Selman proved non-existence of such a reduction for a problem strictly related to CI in [KS92] using an ad-hoc proof.

6 APPLICATIONS

We now apply the theorems presented in the previous section to several PKR formalisms. We assume that definitions of propositional logic, default logic [Rei80], circumscription [McC80], and stable model semantics for logic programs [GL88] are known. Definitions of other less known PKR formalisms follow.

WIDTIO and SBR are two belief revision formalisms. Let K be a set of propositional formulae, representing an agent's knowledge about the world. When a new formula A has to be added to K, the problem of the possible inconsistency between K and A arises. $K * A$ denotes the result of such incorporation. We define:

$$W(K, A) = \{K' \mid K' \text{ is a maximal consistent subset of } K \cup \{A\} \text{ containing } A\}$$

Any $K' \in W(K, A)$ is a maximal choice of formulas in K that are consistent with A and, therefore, we may retain when incorporating A.

SBR (Skeptical Belief Revision [FUV83, Gin86]) The revised theory is defined as a set of theories: $K * A \doteq \{K' \mid K' \in W(K, A)\}$. Logical consequence in the revised theory is defined as logical consequence in each of the theories:

$$K * A \models_{SBR} Q \quad \text{iff} \quad \text{for all } K' \in W(K, A) \text{ we have that } K' \models Q$$

The model semantics is defined as:

$$M \models_{SBR} K * A \quad \text{iff} \quad \text{there exists a } K' \in W(K, A) \text{ such that } M \models K'$$

WIDTIO (When In Doubt Throw It Out [Win90]) A simpler (but somewhat drastical) approach is the so-called WIDTIO, where we retain only the formulae of K that belong to all (maximally consistent) sets of $W(K, A)$. Thus, consequence is defined as:

$$K * A \models_W Q \quad \text{iff} \quad \bigcap W(K, A) \models Q$$

The model semantics of this formalism is defined as:

$$M \models_W K * A \quad \text{iff} \quad M \models \bigcap W(K, A)$$

We also consider the Generalized Closed World Assumption (GCWA) that is a formalism to represent knowledge in a closed world.

GCWA
(Generalized Closed World Assumption [Min82]) The model semantics is defined as (a is a letter):

$$M \models_G KB \quad \text{iff } M \models KB \cup \{\neg a \mid \text{for any positive clause } \gamma, \text{ if } KB \not\models \gamma \text{ then } KB \not\models \gamma \vee a\}$$

Using Theorems 3-6 and results previously known from the literature, we can obtain some new results on model- and theorem-compactness of PKR formalisms. Old and new results on space efficiency of PKR formalisms are presented in Table 1. Results with no reference are new (a dash "–" denotes a folklore result).

First of all, notice that space efficiency is not always related to time complexity. As an example, we compare in detail WIDTIO and circumscription. From the table it follows that both model checking and theorem proving are harder for WIDTIO than for circumscription. Nevertheless, since circumscription is thm-Σ_2^p-complete and WIDTIO is thm-coNP-complete (and thus in thm-Σ_2^p), there exists a poly-size reduction from WIDTIO to circumscription satisfying theorem preservation. The converse does not hold: since circumscription is thm-Σ_2^p-complete and WIDTIO is thm-coNP, there is no theorem-preserving poly-size reduction from the former formalism to the latter. Hence, circumscription is a more compact formalism than WIDTIO to represent theorems. Analogous considerations can be done for models. Intuitively, this is due to the fact that for WIDTIO both model checking and inference require a lot of work on the revised knowledge base alone—computing the intersection of all elements of $W(K, A)$. Once this is done, one is left with model checking and inference in PL. Hence, WIDTIO has the same space efficiency as PL, which is below circumscription.

Figures 3 and 4 contain the same information of Table 1, but highlight existing reductions. Each figure contains two diagrams, the left one showing the existence of polynomial-time reductions among formalisms, the right one showing the existence of poly-size reductions. An arrow from a formalism to another denotes that the former can be reduced to the latter one. We use a bidirectional arrow to denote arrows in both directions and a dashed box to enclose formalisms that can be reduced one into another. Note that some

	Time Complexity		Space Efficiency	
	Model	Theorem	Model	Theorem
Propositional Logic	P –	coNP-complete [Coo71]	model-P –	thm-coNP-complete [CDS95]
WIDTIO	Σ_2^p-complete [LS96]	Π_2^p-hard, in $\Delta_3^p[\log n]$ [EG92]	model-P [CDLS95]	thm-coNP-complete [CDLS95]
Skeptical Belief Revision	coNP-complete [LS96]	Π_2^p-complete [EG92]	model-coNP-complete [LS96]	thm-Π_2^p-complete [CDLS95]
circumscription	coNP-complete [Cad92]	Π_2^p-complete [EG93]	model-coNP-complete [CDSS95]	thm-Π_2^p-complete [CDSS95]
GCWA	coNP-hard, in $\Delta_2^p[\log n]$ [EG93]	Π_2^p-hard, in $\Delta_3^p[\log n]$ [EG93]	model-P [CDSS95]	thm-coNP-complete [CDSS95]
Skeptical Default Reasoning	Σ_2^p-complete [Got92]	Π_2^p-complete [Got92]	model-Σ_2^p-complete	thm-Π_2^p-complete
Credulous Default Reasoning	N/A	Σ_2^p-complete [Got92]	N/A	thm-Σ_2^p-complete
Stable Model Semantics	P –	coNP-complete [MT91]	model-P –	thm-coNP-complete

Table 1: Complexity and Space Efficiency of Formalisms

formalisms are more appropriate in representing sets of models, while others perform better on sets of formulae. An interesting relation exists between Skeptical Default Reasoning and circumscription. While there is no model-preserving poly-size reduction from circumscription to Skeptical Default Reasoning, as already shown in [GKPS95], a theorem-preserving poly-size reduction exists.

7 CONCLUSIONS

In a recent paper [CDLS96] we introduced a new complexity measure, i.e., *compilability*. In this paper we have shown how this measure is inherently related to the succinctness of PKR formalisms. We analyzed PKR formalisms wrt two succinctness measures: succinctness in representing sets of models and succinctness in representing sets of theorems.

We provided a formal way of talking about the relative ability of PKR formalisms to compactly represent information. In particular, we were able to formalize the intuition that a specific PKR formalism provides "one of the most compact ways to represent models/theorems" among the PKR formalisms of a specific class.

Using this tool we have been able to improve on our previous work on compact representations [CDS95, CDSS95, CDLS95] as well as the work by Gogic, Kautz, Papadimitriou and Selman [GKPS95].

Acknowledgments

This work was partially supported by ASI (Italian Space Agency) and CNR (National Research Council of Italy).

References

[BED91] R. Ben-Eliyahu and R. Dechter. Default logic, propositional logic and constraints. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pages 379–385, 1991.

[BED94] R. Ben-Eliyahu and R. Dechter. Propositional semantics for disjunctive logic programs. *Annals of Mathematics and Artificial Intelligence*, 12:53–87, 1994.

[Cad92] M. Cadoli. The complexity of model checking for circumscriptive formulae. *Information Processing Letters*, 44:113–118, 1992.

[CDLS95] 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.

[CDLS96] M. Cadoli, F. M. Donini, P. Liberatore, and M. Schaerf. Feasibility and unfeasibility of off-line processing. In *Proceedings of the Fourth*

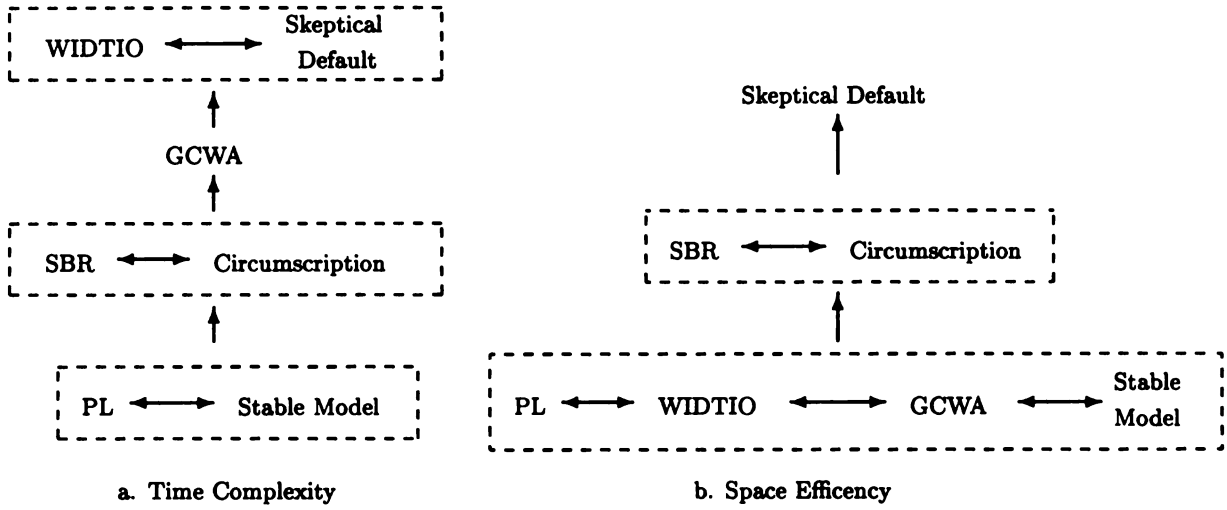


Figure 3: Complexity of Model Checking vs. Space Efficiency of Model Representation

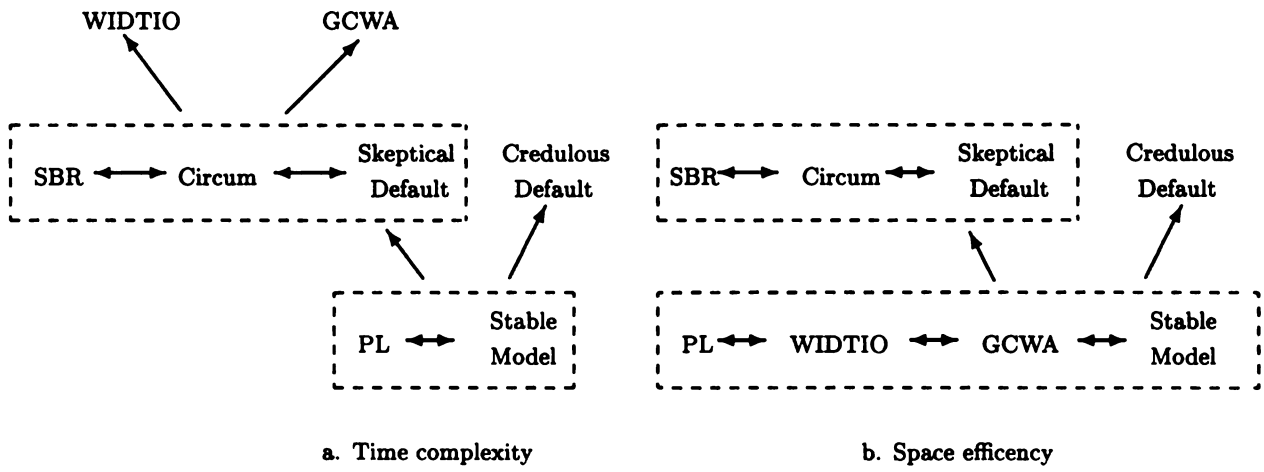


Figure 4: Complexity of Inference vs. Space Efficiency of Theorem Representation

Israeli Symposium on Theory of Computing and Systems (ISTCS-96), pages 100–109, 1996.

[CDS95] M. Cadoli, F. M. Donini, and M. Schaerf. Is intractability of non-monotonic reasoning a real drawback? Technical Report RAP.09.95, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", July 1995. To appear in *Artificial Intelligence Journal*. Short version appeared in *Proc. of AAAI-94*, pages 946–951.

[CDSS95] M. Cadoli, F. M. Donini, M. Schaerf, and R. Silvestri. On compact representations of propositional circumscription. Technical Report RAP.14.95, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", July 1995. To appear in *Theoretical Computer Science*. Short version appeared in *Proc. of STACS-95*, pages 205–216.

[Coo71] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third ACM Symposium on Theory of Computing (STOC-71)*, pages 151–158, 1971.

[EG92] T. Eiter and G. Gottlob. On the complexity of propositional knowledge base revision, updates and counterfactuals. *Artificial Intelligence Journal*, 57:227–270, 1992.

[EG93] T. Eiter and G. Gottlob. Propositional circumscription and extended closed world reasoning are Π_2^p-complete. *Theoretical Computer Science*, pages 231–245, 1993.

[Eth87] D. V. Etherington. *Reasoning with incomplete information*. Morgan Kaufmann, Los Altos, Los Altos, CA, 1987.

[FUV83] R. Fagin, J. D. Ullman, and M. Y. Vardi. On the semantics of updates in databases. In *Proceedings of the Second ACM SIGACT SIGMOD Symposium on Principles of Database Systems (PODS-83)*, pages 352–365, 1983.

[Gin86] M. L. Ginsberg. Conterfactuals. *Artificial Intelligence Journal*, 30:35–79, 1986.

[GJ79] 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.

[GKPS95] G. Gogic, H. 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.

[GL88] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings of the Fifth Logic Programming Symposium*, pages 1070–1080. The MIT Press, 1988.

[Got92] G. Gottlob. Complexity results for nonmonotonic logics. *Journal of Logic and Computation*, 2:397–425, 1992.

[GPP89] M. Gelfond, H. Przymusinska, and T. Przymusinsky. On the relationship between circumscription and negation as failure. *Artificial Intelligence Journal*, 38:49–73, 1989.

[Joh90] 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.

[KL80] 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.

[KS92] 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.

[Lib95] P. Liberatore. Compact representation of revision of Horn clauses. In *Proceedings of the Eighth Australian Joint Artificial Intelligence Conference*, 1995.

[LS96] P. Liberatore and M. Schaerf. The complexity of model checking for belief revision and update. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, 1996. To appear.

[McC80] J. McCarthy. Circumscription - A form of non-monotonic reasoning. *Artificial Intelligence Journal*, 13:27–39, 1980.

[Min82] J. Minker. On indefinite databases and the closed world assumption. In *Proceedings of the Sixth International Conference on Automated Deduction (CADE-82)*, pages 292–308, 1982.

[MT91] W. Marek and M. Truszczyński. Autoepistemic logic. *Journal of the ACM*, 38(3):588–619, 1991.

[Rei80] R. Reiter. A logic for default reasoning. *Artificial Intelligence Journal*, 13:81–132, 1980.

[Sto76] L. J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1976.

[Win89] M. Winslett. Sometimes updates are circumscription. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 859–863, 1989.

[Win90] M. Winslett. *Updating Logical Databases*. Cambridge University Press, 1990.

[Yap83] C. K. Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical Computer Science*, 26:287–300, 1983.

Encoding Plans in Propositional Logic

Henry Kautz, David McAllester, and Bart Selman
AT&T Labs
600 Mountain Avenue
Murray Hill, NJ 07974
{kautz, dmac, selman}@research.att.com

Abstract

In recent work we showed that planning problems can be efficiently solved by general propositional satisfiability algorithms (Kautz and Selman 1996). A key issue in this approach is the development of practical reductions of planning to SAT. We introduce a series of different SAT encodings for STRIPS-style planning, which are sound and complete representations of the original STRIPS specification, and relate our encodings to the Graphplan system of Blum and Furst (1995). We analyze the size complexity of the various encodings, both in terms of number of variables and total length of the resulting formulas. This paper complements the empirical evaluation of several of the encodings reported in Kautz and Selman (1996). We also introduce a novel encoding based on the theory of *causal* planning, that exploits the notion of "lifting" from the theorem-proving community. This new encoding strictly dominates the others in terms of asymptotic complexity. Finally, we consider further reductions in the number of variables used by our encodings, by compiling away either state-variables or action-variables.

1 INTRODUCTION

In recent work we have shown that planning problems can be efficiently solved by general propositional satisfiability algorithms (Kautz and Selman 1996). On instances drawn from logistics and blocks-world planning problems, both systematic and local-search SAT algorithms often dramatically outperform earlier domain-independent planning systems. A key issue in this approach is the development of practical reductions of planning to SAT. Because both SAT and bounded-length STRIPS-style planning are NP-complete, it is evident that polynomial reductions between the problems exist. However, there are both relative and absolute practical limits on the size complexity of an acceptable reduction. For example, an $O(n^3)$ increase in the size of problem may be acceptable, while a reduction with an $O(n^4)$ may yield SAT instances that are too large to be solved by current algorithms. Of course, not all SAT problems of a given size are equally difficult, and the known results that quantify the hardness of randomly-generated SAT problems based on its size and clause-ratio (Mitchell, Selman, and Levesque 1992) cannot be directly applied to structured problems. However, experiments reported in Kautz and Selman (1996) using the SATPLAN system on problems derived from planning provide some rough guidelines. Formulas containing around 2,000 variables could be solved by both systematic and stochastic search in a few seconds. The limits of the systematic algorithm ("tableau", Crawford and Auton (1993)) were reached at 2,800 variables and 6 hours of running time. For the stochastic algorithm ("Walksat", Selman *et al.* (1994; 1996)), problems containing 6,700 variables were solved in 15 minutes; informal experiments show that the stochastic methods are currently feasible up to around 10,000 or so variables.

Kautz and Selman (1992) described one possible reduction of blocks-world planning to SAT, and some associated encoding techniques for reducing the number of variables in the SAT encoding. Kautz and Selman (1996) also showed that the "planning graphs" used by Blum and Furst's Graphplan system could be interpreted as propositional CNF formulas, and informally described an alternative reduction of planning to SAT called "state-based encodings". Both of these papers, however, concentrated on experiments in actually solving the resulting SAT formulas. In this paper, we focus on the reductions themselves. We present a analysis of the size of the encoded problems, noting the degree to which various properties of the original STRIPS problem (*e.g.*, the number of operators, the size of the domain, the length of the minimal solution, *etc.*) affect the size of the encoding under different reductions. This kind of analysis and comparison allows the gross statistical properties of the statement of a given planning problem to be used to select an efficient encoding.

The first reduction, "linear encodings", is most reminiscent of classical situation calculus representations of planning. Second, we show how the use of (the SAT equivalent of) "explanatory" frame axioms (Haas 1987; Schubert 1989)

can both reduce the problem size and allow parallel actions to be efficiently encoded, in a manner very similar to the planning graphs of Blum and Furst. Two related encoding techniques, "operator splitting" and "lifting", are described, that reduce the number of variables required in the SAT reduction. Third, we present a new encoding of planning into SAT based on the lifted version of "causal plans" introduced by McAllester and Rosenblitt (1991). Lifted casual plan encodes have the best size complexity in the limit (although possibly with a larger constant factor).

Finally, we consider ways to reduce the size of an encoded problem by "compiling away" certain classes of variables. Because resolution can be used to eliminate any set of variables from a SAT problem, our encodings can be further transformed so that either they contain no variables referring to actions, or no variables referring to states. The first transformation gives an automated procedure for creating the "state-based encodings" that were created manually for the experiments reported in Kautz and Selman (1996). For the Graphplan system, we show that the worst-case size of the transformed formula is strictly better when state-variables are eliminated, rather than action-variables. However, these worst-case results should be tempered by the fact that we have, by hand, created state-based encodings of *particular* domains (*e.g.* logistics planning) that are much more compact than those given by general reductions. The results in this paper are dealt with in more detail in Kautz *et al.* (1996).

While this paper concentrates on reductions of planning to Boolean satisfiability, it is important to note that there is related work that views planning as general constraint satisfaction (*e.g.*, Joslin and Pollack 1995). We believe that the various techniques introduced in our reductions, such as operator splitting, lifting, and compilation, will also prove to be applicable in the more general CSP setting. The operation of our SATPLAN system can also be viewed as a form of refinement planning, as shown by Kambhampati and Yang (1996).

2 DEFINITIONS AND GENERAL FRAMEWORK

Planning problems are specified using standard STRIPS notation (Fikes and Nilsson 1971). A planning problem Π is a tuple $\langle \text{Ops}, \text{Dom}, S_0, S_1 \rangle$, where Ops is a set of operator definitions, Dom is a domain of individuals, and S_0 and S_1 are the initial and goal states, respectively. Operators are defined by schemas using precondition, add, and delete lists; for example:

```
MOVE(x, y, z)
  PRE: CLEAR(x), ON(x, y), CLEAR(z)
  ADD: CLEAR(y), ON(x, z)
  DEL: CLEAR(z), ON(x, y)
```

All of the variables that appear in the definition of an operator must appear in its head. (In particular, note that no negated atoms appear anywhere in this formalism.) The instantiation of an operator over the domain is called an *action* (*e.g.*, `MOVE(A,B,C)`), and the instantiation of a predicate is called a *fluent* (*e.g.*, `ON(A,B)`).[1] Given an operator definition, it is straightforward to define functions Pre(), Add(), and Del() that map an operator or an action to the corresponding lists of predicates or fluents; *e.g.*, Add(`MOVE(A,B,C)`) = {`ON(A,C)`, `CLEAR(B)`}. States are sets of fluents.

An action applied to a state adds or deletes the specified fluents from the state. A sequence of actions is a *solution* to a planning problem if it transforms the initial state into a superset of the goal state (the goal need only be partially specified), the precondition of each action appears in the state to which it applies, and no action both adds and deletes the same fluent. In a *bounded* planning problem, the solution must be length $\leq n$ for some fixed n. Given this notion of a planning problem, we can then define what it means to reduce a planning problem to SAT.

Definition: A function $\mathcal{T}$ which takes a planning problem Π and a length bound n and returns a SAT problem is said to *reduce bounded planning to SAT* provided that $\mathcal{T}(\Pi, n)$ is solvable by a plan with of at most n operations if and only if $\mathcal{T}(\Pi, n)$ is satisfiable. A SAT embedding is called constructive if a solution to Π can be efficiently extracted from any solution to $\mathcal{T}(\Pi, n)$ — more specifically if there exists a polynomial time operation $\mathcal{E}$ such that for any Π, n, and truth assignment σ satisfying $\mathcal{T}(\Pi, n)$, we have that $\mathcal{E}(\Pi, n, \sigma)$ is a solution to Π.

Each of the reductions described below is constructive. All the reductions are to CNF, which is the form used by most satisfiability-testing programs. (For clarity in the exposition we sometimes write down formulas which are not strictly CNF, but which can obviously put into CNF form with only a linear increase in size.) Whenever we talk of the "size" of a formula, we mean the total number of literals it contains. When we are considering clauses that are of fixed length, we can also talk about size in terms of the number of clauses without confusion.

For each reduction we also analyze the size of the CNF SAT encoding, as a function of various features of the planning problem instance. These features include the number of operators |Ops|, predicates |Pred|, domain elements |Dom|, and length bound n. The linear and Graphplan based reductions are also critically dependent on maximum number of arguments (*i.e.*, arity) of the operators and predicates, noted as $\mathcal{A}_{\text{Ops}}$ and $\mathcal{A}_{\text{Pred}}$ respectively. (The size of the lifted causal encoding is independent of the arity of the operators and predicates.) We abbreviate the expression for the number of actions, namely $|\text{Ops}||\text{Dom}|^{\mathcal{A}_{\text{Ops}}}$, as $|\mathcal{A}|$. Similarly, we abbreviate the number of fluents, $|\text{Pred}||\text{Dom}|^{\mathcal{A}_{\text{Pred}}}$, as

[1] For simplicity we assume a single domain over which all variables range, but it is easy to extend the definitions to allow typed domains and variables.

$|\mathcal{F}|$. Finally, we use m to denote the maximum combined length of the precondition, add, and delete lists for any operator.

3 LINEAR ENCODINGS

Linear encodings, introduced by Kautz and Selman (1992), are quite similar to a propositionalized version of the situation calculus (McCarthy and Hayes 1969), but with additional axioms included to rule out certain "unintended models" that are harmless for deductive formalizations, but problematic for the model-finding (SAT) approach. The intuition behind the encoding is that an additional time-index parameter is added to each action or fluent, to indicate the state at which the action begins or the fluent holds; for a problem bounded by n, the actions are indexed by 0 through $n-1$, and the fluents by 0 through n. Variables corresponding to dummy "null" action are also included: these handle the case where the solution to the planning problem is actually shorter than n. In brief, the reduction yields the following kinds of clauses:

1. The initial state is completely specified: if fluent $f \notin S_0$, then $\neg f_0$.[2] The goal state may be either fully or partially specified.
2. If an action holds at time i, its preconditions hold at i, its added fluents hold at $i+1$, and the negation of each of its deleted fluents holds at $i+1$.
3. Classical frame conditions hold for all actions (*i.e.*, if an action does not add or delete a fluent, then the fluent remains true or remains false when the action occurs).
4. Exactly one action occurs at each time instant (exclusiveness).

The correctness of the embedding can be shown by induction on n: propositions indexed by 0 exactly correspond to the given initial state, and those indexed by $i+1$ describe a legal state that can be reached from one described by propositions indexed by i. The function $\mathcal{E}$ simply extracts the sequence of action instances that are true in a satisfying truth assignment.

The number of variables used is $O(n|\mathcal{A}| + n|\mathcal{F}|)$. The size of the CNF formula (number of literal occurrences) is dominated by the clauses corresponding to the exclusiveness and frame axioms, and is thus $O(n|\mathcal{A}|^2 + n|\mathcal{A}||\mathcal{F}|)$. The most critical factor in determining whether this reduction is practical is clearly the arity of the operators and predicates. For many planning domains the predicate arity is 2 (*e.g.*, ON, IN, and NEXT-TO). Operators that take 3 or more arguments (*e.g.* MOVE) generally make this reduction infeasible. We therefore consider two modifications to this reduction that shrink the number of variables and clauses.

[2]Note that this encoding makes an explicit closed-world assumption about the initial state. The STRIPS formalism itself does not necessarily require this assumption: STRIPS can be consistently interpreted so that the absence of a fluent from a state means that it is unknown, rather than false. However, this subtle difference does not actually change the set of solutions to a given problem, due to the restricted form of the operator definitions.

3.1 Operator Splitting

The first modification, operator splitting, is based on the observation that since only a single action occurs at a given time step, an m-place action can be represented as the conjunction of m 1-place actions. For example, instead of encoding "the action MOVE(A,B,C) occurs at time 3" as a single proposition "MOVE(A,B,C,3)", one could use an expression like "SOURCE(A,3) ∧ OBJECT(B,3) ∧ DESTINATION(C,3)". This technique can also be viewed as a special case of a "lifted" representation, that is, the use of propositions to represent bindings of the arguments of an operator or predicate. Lifting is used more extensively in the causal encodings described below. Operator splitting reduces the number of variables to $O(n\mathcal{A}_{\text{Ops}}|\text{Ops}||\text{Dom}| + n|\mathcal{F}|)$. The exclusiveness axioms become small, requiring only $O(n\mathcal{A}_{\text{Ops}}{}^2|\text{Ops}|^2|\text{Dom}|^2)$ binary clauses, because one can separately assert that each of the 1-place actions has a unique argument. Furthermore, in many cases operator splitting reduces the number of clauses corresponding to frame axioms, because not all arguments to an action may need appear in a frame axiom. Using this modification to the basic linear encodings, Kautz and Selman (1996) were able to solve blocks-world problems requiring 20 blocks and 20 moves, without using any kind of domain-specific search control knowledge.

3.2 Explanatory Frame Axioms

Haas (1987) and Schubert (1989) proposed an alternative to McCarthy and Hayes' style frame axioms, in the context of first-order deductive formalizations of planning. An "explanatory" frame axiom says that if the truth value of a fluent changes, then one of the actions that adds or deletes it must have occured. If none of those actions occurs, then by *modus tolens* the truth value of the fluent must persist through whatever other action does occur. These kind of frame axioms can be incorporated directly into the linear encoding framework. The basic schemas (which can be expanded into a set of clauses) are:

$$f_i \wedge \neg f_{i+1} \supset \bigvee\{a_i | f \in \text{Del}(a_i)\}$$

$$\neg f_i \wedge f_{i+1} \supset \bigvee\{a_i | f \in \text{Add}(a_i)\}$$

While classical frame axioms require $O(n|\mathcal{A}||\mathcal{F}|)$ clauses, this modification uses only $O(n|\mathcal{F}|)$ clauses. Unfortunately, each clause may be longer, and in the worst case, the total size of the formula is the same. However, in practice it appears that this formulation leads to smaller encodings, because the number of actions that could explain a given change is usually small, and so the clauses are of moderate length. Thus, if at most k actions could account for a change in the truth of a fluent, then the total size of the frame axioms is $O(nk|\mathcal{F}|)$.

When both operator splitting and explanatory frame axioms are employed, the size of the entire reduction is then dominated by the size of the frame axioms. We will discuss this issue in the expanded version of this paper (Kautz *et al.* 1996).

When explanatory frame axioms are used, one may optionally weaken the exclusiveness axioms, so that they assert that at *most* one action occurs at each time instance. Furthermore, it is no longer necessary to introduce "null" actions to account for solutions that are shorter than n. This is because the explanatory axioms imply that an action occurs whenever the (encodings of) two adjacent states differ. Thus, if two adjacent states are identical, then no action occurs at that instant.

4 PARALLELIZED ENCODINGS

So far, we have considered encodings in which only one action can occur at each time step. The number of time steps n occurs as a factor in both the number of variables and the total length of the encodings. An obvious way therefore to reduce the size of our encodings is by allowing several actions to occur at each time step, *i.e.*, parallel actions. The encodings below are such that if several actions occur at the same time, it means that they can be serialized in any order. Therefore the solution extraction function $\mathcal{E}$ for these encodings would identify the partially-ordered set of actions in a satisfying model and then arrange the actions in an arbitrary total order.

4.1 Graphplan-based Encodings

The Graphplan system of Blum and Furst (1995) works by converting a STRIPS-style specification into a planning graph. This is an ordered graph, where alternating layers of nodes correspond to grounds facts (indexed by the time step for that layer) and fully-instantiated operators (again indexed by the time step). Arcs lead from each fact to the operators that contain it as a precondition in the next layer, and similarly from each operator to its effects in the next layer. For every operator layer and every fact there is also a no-op "maintain" operator that simply has that fact as both a precondition and "add" effect.

A solution is a subgraph of the planning graph that contains all the facts in the initial and goal layers, and contains no two operators in the same layer that conflict (*i.e.*, one operator deletes a precondition or an effect of the other). Thus, a solution corresponds to a partially-ordered plan, which may contain several operators occuring at the same time step, with the semantics that those operators may occur in any order (or even in parallel). For planning problems that can take advantage of this kind of parallelism, the planning graph can have many fewer layers than the number of steps in a linear solution — and therefore be much smaller.

A planning graph is quite similar to a propositional formula, and in fact, we can automatically convert planning graphs into CNF notation. The translation begins at goal-layer of the graph, and works backward. Using the "rocket" problem in Blum and Furst (1995, Fig. 2) as an example (where "`LOAD(A,R,L,i)`" means "load `A` into `R` at location `L` at time `i`", and "`MOVE(R,L,P,i)`" means "move `R` from `L` to `P` at time `i`"), the translation is:

1. The initial state holds at layer 1 (fully specified), and the goals hold at the highest layer.
2. Operators imply their preconditions; *e.g.*,
$$\texttt{LOAD(A,R,L,2)} \supset (\texttt{AT(A,L,1)} \wedge \texttt{AT(R,L,1)}$$
3. Each fact at level i implies the disjunction of all the operators at level $i-1$ that have it as an add-effect (backward-chaining); *e.g.*,
$$\texttt{IN(A,R,3)} \supset (\texttt{LOAD(A,R,L,2)} \vee \texttt{LOAD(A,R,P,2)} \vee \texttt{MAINTAIN(IN(A,R),2)})$$
4. Conflicting actions are mutually exclusive; *e.g.*,
$$\neg\texttt{LOAD(A,R,L,2)} \vee \neg\texttt{MOVE(R,L,P,2)}$$

The axioms of type (2) above assert that actions imply their preconditions, but not that actions imply their effects. This can admit solutions that contain spurious actions. The extraction function $\mathcal{E}$ can simply delete actions from the solution whose effects do not hold. The axioms for conflicting actions (4) prevent the solution from actually *depending* upon the co-occurance of actions whose effects would conflict.

The resulting CNF formula has $O(n|\mathcal{A}| + n|\mathcal{F}|)$ variables. The size of the formula is given by $O(nm|\mathcal{A}| + n|\mathcal{A}|^2 + nk|\mathcal{F}|)$. Note that is the same as the expression for the size of our linear encoding with explanatory frame axioms, but *without* operator splitting. This is consistent with the empirical results reported in Kautz and Selman (1996), where we observed that on the larger benchmark instances, the Graphplan-based encodings became too large for our satisfiability procedures. It should also be noted that these are worst-case bounds. In the worst-case, all operators are mutually exclusive, and therefore, we do not see an advantage in allowing parallel actions. In many practical domains, however, parallel actions lead to fewer "exclusiveness" axioms, fewer time steps (smaller n), and thus shorter encodings.

A very similar encoding for plans with parallelized actions can be generated by using the axioms for linear encodings with explanatory frame axioms (Section 3.2), but where the exclusiveness axioms only assert that conflicting actions are mutually exclusive, as in the Graphplan-based encoding. In fact, explanatory frame axioms can be generating by resolving together the backward-chaining axioms (3) above with the action/precondition axioms (2) for the "maintain" actions. For example, resolving the backward-chaining axiom:
$$\texttt{IN(A,R,3)} \supset (\texttt{LOAD(A,R,L,2)} \vee \texttt{LOAD(A,R,P,2)} \vee \texttt{MAINTAIN(IN(A,R),2)})$$

with the action/precondition axiom:

$$\texttt{MAINTAIN(IN(A,R),2)} \supset \texttt{IN(A,R,1)}$$

yields the the frame axiom:

$$(\neg\texttt{IN(A,R,1)} \wedge \texttt{IN(A,R,3)}) \supset (\texttt{LOAD(A,R,L,2)} \vee \texttt{LOAD(A,R,P,2)})$$

4.2 Compiling Away Actions or Fluents

Our linear and Graphplan encoding have a special structure in terms of the dependencies among variables. More specifically, we have alternating layers of variables representing operators and variables representing states (fluents). Since we are interested in reducing the number of variables as much as possible, we now consider whether we can "compile away" the action layers or the state layers. Given two consecutive states, it's generally straightforward to determine what action(s) led to the changes between the states. The action variables can thus be viewed as logically dependent variables. Similarly, the state variables can be taken as the dependent ones, since a set of actions applied at a certain state directly defines the changes in that state. So, we can compile away the dependent variables: either the state variables or the action variables.

First, note that for any propositional theory, *any* variable can be eliminated by performing all possible resolutions on that variable, and then removing all clauses containing the variable. Thus, any subset of propositions, whether they represent actions or fluents, can in principle be eliminated. The key question is what happens to the total number of clauses and the size of the clauses in the encodings. In general the resolution procedure gives an exponential blowup in size. For Graphplan, however, we know the following:

> **Observation:** For Graphplan-based encodings, compiling away actions can lead to exponential blowup in the size of the encoding, but compiling away states gives only a polynomial (in |Dom|) increase in size.

Consider the result of resolving clauses between action- and state- layers. When resolving away action variables, we find that a given fluent, say, s_2, can have been added by a number of different actions, *e.g.*, a_1^1, a_1^2, or a_1^3. Each of these actions may have a different pair of pre-conditions, *e.g.*, a_1^1 may require s_1^1 and s_1^2, while a_1^2 requires s_1^3 and s_1^4, etc. We get $s_2 \supset (a_1^1 \vee a_1^2 \vee a_1^3)$. This gives us $s_2 \supset ((s_1^1 \vee s_1^2) \vee (s_1^3 \vee s_1^4) \vee (s_1^5 \vee s_1^6))$. Going back to CNF, we get clauses of the form $s_2 \supset (s_1^1 \vee s_1^3 \vee s_1^5)$, $s_2 \supset (s_1^2 \vee s_1^3 \vee s_1^5)$, etc. In general, we get exponentially many, *i.e.*, a worst-case blowup of $O(|\mathcal{F}|^{|\mathcal{A}|})$. More precisely, the blowup is $O(|\mathcal{F}|^k)$, where k is maxium number of actions that could account for a change in the truth of a fluent.

However, to compile away states, *e.g.*, s_2^1, s_2^2, you simply go from $a_2^1 \supset s_2^1$, $a_2^1 \supset s_2^2$, $a_2^2 \supset s_2^1$, $a_2^2 \supset s_2^2$, and $s_2^1 \supset (a_1^1 \vee a_1^2)$ to $a_2^1 \supset (a_1^1 \vee a_1^2)$, $a_2^2 \supset (a_1^1 \vee a_1^2)$, which gives a total worst case increase of $O(|\mathcal{A}|)$.

An intuitive explanation for the fact that Graphplan-based representations do not blow-up when explicit variables are eliminated is that the dummy "maintain" actions implicitly encode state information.

Encodings with the actions compiled away were called *state-based* encodings in Kautz and Selman (1996). For a further discussion of state-based encodings see Bendrax-Weiss *et al.* (1996). Encodings resulting from compiling away states, leaving "action-based " encodings, are similar to the causal encodings considered in McAllester and Rosenblitt (1991), Penberthy and Weld (1992), and Barrett and Weld (1994). Our analysis shows that the worst-case size of the action-based encoding is strictly better than that of the the state-based encodings. It should be noted however that in Kautz and Selman (1996), we still obtained good experimental results with state-based encodings. This is because in the domains considered, the number of actions that could account for a change in the truth of a fluent was small. It therefore appears that in this case our worst-case results may be overly pessimistic for many domains.

One cannot show in general that state-based encodings are constructive, because it may be difficult to reconstruct the set of actions that admit the sequence of states represented by models of the axioms. The general problem of finding unordered plans of length 1 is NP-complete. However, in domains we have examined so far, including the logistics and blocks world domains, there is a linear-time algorithm for finding such plans – again, because there are few ways of changing the truth-value of a single fluent.

5 LIFTED CAUSAL ENCODINGS

We now give a reduction inspired by the lifted version of the SNLP causal link planner of McAllester and Rosenblitt (1991). This encoding gives our best asymptotic results. Unlike our other reductions, which are exponential in $\mathcal{A}_{\text{Ops}}$ and $\mathcal{A}_{\text{Pred}}$, this reduction is polynomial in all parameters. The analysis can be simplified by considering a fixed but arbitrary set of operators. Furthermore, we consider the case where $|S_0| = |S_G| = |\text{Dom}| = n$. This is reasonable in the blocks world where the size of the initial and final state, the number of blocks, and the plan length are all roughly linearly related. Under these conditions (with an arbitrary operator set) this encoding gives a SAT formula with $O(n^2)$ Boolean variables and $O(n^3)$ literal occurrences.

We will describe a lifted causal solution to the Sussman anamoly in three steps. First we review ground nonlinear causal planning and discuss a reduction to SAT based on the ground case. We then introduce the notion of a lifted SAT problem, an NP-complete problem somewhat "more general" than SAT. Every SAT problem is a lifted SAT problem but not vice-versa. We then give a reduction from planning to lifted SAT. Finally we give a general reduction from lifted SAT to SAT.

We will use the Sussman anomaly as an explanatory example. Suppose that we have one operator definition for the

MOVE operation as follows.

```
MOVE(x, y, z)
  PRE: CLEAR(x), ON(x, y), CLEAR(z)
  ADD: CLEAR(y), ON(x, z)
  DEL: CLEAR(z), ON(x, y)
```

Suppose that we have block C on block A, block A on PLACE1; block B on PLACE2, with C, B, and PLACE3 clear. Suppose we want A on B on C. The shortest plan is to move C from A to PLACE3; move B from PLACE2 to C; and finally move A from PLACE1 to B.

In this formulation of the Sussman anomaly we have six objects (three blocks and three places). This yields 6^3, or 216 possible actions of the form MOVE(x, y, z). In this formulation of the blocks world we have $|\text{Dom}|^3$ possible move actions. Each move action has three prerequisites, two deletions and two additions. A ground planner works with the individial ground actions and ignores the general operator definition.

We can think of the initial state as an operation which adds the initial assertions and the final state as an operation which requires, as prerequisites, the final assertions. The basic principle of causal link planning is that every prerequisite, including the assertions in the final state, must have a causal source, which may be the initial state. In the Sussman anomaly there are two assertions required by the final state and three required by each of the three actions. So we have 11 total prerequisites. A ground causal link is an assertion of the form $o_i \xrightarrow{\Phi} o_j$ where o_i and o_j are plan steps (possibly the initial or final step), and Φ is a ground fluent that is a prerequisite of o_j. The causal link assertion is true if o_i adds Φ, o_j needs Φ as a prerequisite and no step between o_i and o_j either adds or deletes Φ.

A nonlinear ground causal planner works with a set of step names where two different step names might be assigned the same action (some plans require the same action to be performed twice at different times). We let $o_1, \ldots, o_n$ be the step names. There is no *apriori* temporal order on the step names, and assertions of the form $o_i < o_j$ are used to state that step o_i occurs before step o_j.

A complete causal plan is an assignment of ground actions to step names, a set of casual links, and a set of step ordering assertions satisfying the following conditions.

1. Every prerequisite has a cause. If Φ is a prerequisite of the action assigned to step o_i then the plan includes a causal link of the form $o_j \xrightarrow{\Phi} o_i$.

2. Every causal link is true. If the plan contains $o_j \xrightarrow{\Phi} o_i$ then the action assigned to o_j adds Φ, the plan contains the ordering assertion $o_j < o_i$, and for every step name o_k other than o_i and o_j such that the action assigned to o_k deletes Φ, the plan contains either $o_k < o_j$ or $o_i < o_k$.

3. The ordering constraints are consistent. If the ordering constraints contain $o_i < o_k$ and $o_k < o_j$ then they contain $o_i < o_j$ and no step precedes itself, *i.e.*, the ordering constraints do not contain $o_i < o_i$ for any step o_i.

Theorem: In any topological sort of a complete nonlinear clausal plan, *i.e.*, in any total order consistent with the ordering constraints, all prerequisites are true when executed.

Recall that the initial and final states can be modeled as initial and final steps in the plan. For the Sussman anomaly the three most significant causal links are the following:

$$\text{MOVE(B,PLACE2,C)} \xrightarrow{\text{ON(B, C)}} \text{FINAL}$$
$$\text{MOVE(A,PLACE1,B)} \xrightarrow{\text{ON(A, B)}} \text{FINAL}$$
$$\text{MOVE(C,A,PLACE3)} \xrightarrow{\text{CLEAR(A)}} \text{MOVE(A,PLACE1,B)}$$

Since there are a total of 11 prerequisites, there are 8 other casual links with the initial state as the source. Also note that moving B onto C deletes CLEAR(C), a prerequisite of MOVE(C,A,PLACE3). The definition of a complete plan forces the step moving B to C to occur after the step moving C to A. Hence, once the actions and causal links are selected in the Sussman anomaly, the ordering of the steps is forced.

Condition 2 above does not ensure systematicity, *i.e.*, that every solution corresponds to a unique complete causal plan. To ensure systematicity we must require that actions that *add* the fluent in a causal link are also ordered to occur outside the link. We assume that systematicity is not important in local search.

We can reduce ground causal nonlinear planning directly to SAT, although this turns out not to be nearly as concise as the lifted encoding. We let I and F be initial and final actions. Let A be the set of actions *not* including actions representing the final or initial state. We let O be the set of step names not including initial and final steps. We let $\mathcal{F}$ be the set of all ground fluents. The ground clauses consist of the clauses in the conjunctive normal form of the Boolean formulas in Table 1.

We give a complexity analysis based on the assumption that the ground planning problem is derived from a fixed set of lifted operator definitions. This implies that each action has a bounded number of preconditions, additions, and deletions. For a fixed set of operation definitions we have that $|A|$ is $O(|\text{Dom}|^{A_{\text{Ops}}})$ and $|\mathcal{F}|$ is $O(|\text{Dom}|^{A_{\text{Pred}}} + |S_0| + |S_G|)$. The number of Boolean variables is dominated by the causal links, which is $O(n^2|\mathcal{F}|)$, and the number of variables of the form ACTION(o, a), which is $O(n|A|)$. The number of literal occurances is dominated by schema 10 which involves $O(n^3|\mathcal{F}|)$ clauses (and literal occurances). For the blocks world with the three place move operation, and with $|\text{Dom}| = |S_0| = |S_G| = n$ (where n is the number

1.	$\texttt{ACTION}(o, a_1) \vee \ldots \vee \texttt{ACTION}(o, a_m)$	$o \in O,\ a_i \in A$
2.	$\neg(\texttt{ACTION}(o, a) \wedge \texttt{ACTION}(o, b))$	$o \in O,\ a, b \in A,\ a \neq b$
3.	$\neg\texttt{ADDS}(I, \Phi)$	$\Phi \in \mathcal{F} - S_o$
4.	$\texttt{NEEDS}(F, \Phi)$	$\Phi \in S_G$
5.	$\texttt{ADDS}(o, \Phi) \equiv (\texttt{ACTION}(o, a_1) \vee \ldots \vee \texttt{ACTION}(o, a_k))$	$o \in O,\ \Phi \in \text{Add}(a_i)$
6.	$\texttt{DELS}(o, \Phi) \equiv (\texttt{ACTION}(o, a_1) \vee \ldots \vee \texttt{ACTION}(o, a_k))$	$o \in O,\ \Phi \in \text{Del}(a_i)$
7.	$\texttt{NEEDS}(o, \Phi) \equiv (\texttt{ACTION}(o, a_1) \vee \ldots \vee \texttt{ACTION}(o, a_k))$	$o \in O,\ \Phi \in \text{Pre}(a_i)$
8.	$\texttt{NEEDS}(p, \Phi) \Rightarrow (o_1 \stackrel{\Phi}{\rightarrow} p \vee \ldots \vee o_n \stackrel{\Phi}{\rightarrow} p)$	$a_i \in O \cup \{I\},\ p \in O \cup \{F\},\ \Phi \in \mathcal{F}$
9.	$o \stackrel{\Phi}{\rightarrow} p \Rightarrow \texttt{ADDS}(o, \Phi)$	$o \in O \cup \{I\},\ p \in O \cup \{F\},\ \Phi \in \mathcal{F}$
10.	$o \stackrel{\Phi}{\rightarrow} p \wedge \texttt{DELS}(r, \Phi) \Rightarrow (r < o \vee p < r)$	$o \in O \cup \{I\},\ p \in O \cup \{F\},\ r \in O - \{o, p\},\ \Phi \in \mathcal{F}$
11.	$I < o$	$o \in O$
12.	$o < F$	$o \in O$
13.	$\neg(o < o)$	$o \in O \cup \{I, F\}$
14.	$o < p \wedge p < r \Rightarrow o < r$	$p, q, r \in O \cup \{I, F\}$

Table 1: Reduction of ground causal nonlinear planning (non-lifted).

of plan steps) we get $O(n^4)$ Boolean variables and $O(n^5)$ literal occurrences.

The causal encoding eliminates the need for frame axioms. The frame axioms are implicit in schema 10. The lifted version of schema 10 involves $O(n^3)$ clauses instead of $O(n^5)$.

Now we define the notion of a lifted SAT problem. A lifted clause is just a first order clause — a disjunction of first order literals possibly containing free variables. A lifted SAT problem is just a set of lifted clauses. A lifted SAT problem is *satisfiable* if there exists a ground substitution (mapping from variables to ground terms) such that the resulting ground SAT problem is satisfiable, subject to the constraint that a ground atomic equality of the form $t = w$ is true if and only if t and w are the same term. As an example, consider the following simple clause set.

$$P(x) \Rightarrow Q(x)$$

$$Q(x) \Rightarrow W(x)$$

$$P(y)$$

$$x = a \vee x = b$$

$$P(a),\quad \neg W(a),\quad P(b),\quad \neg W(b)$$

This lifted SAT problem is not satisfiable. x must be either a or b, and in either case one of the first two clauses must be violated. However, if we remove the second clause which intuitively defines the range of x, then the clause set becomes satisfiable because we can now interpret x as some new constant, say c, and set $P(c)$, $Q(c)$, and $W(c)$ all to true. Note that the clause $P(y)$, which intuitively constrains the value of y, has no effect on the satisfiability of the problem. This existential interpretation of the variables in the clauses is very different from the universal interpretation assigned to clauses in resolution theorem proving.

Theorem: A lifted SAT problem is satisfiable if and only if there exists an equivalence relation on the atomic formulas and a truth assignment to the atomic formulas respecting that relation (equivalent literals are assigned the same truth value), such that the truth assignment satisfies all clauses and the unification problem defined by the equivalence relation is solvable.

Proof: If there exists such a truth assignment and equivalence relation then the most general unifier of the equivalences in the equivalence relation yields a substitution satisfying the problem. On the other hand, any substitution and truth assignment satisfying the problem defines an equivalence relation and truth assignment satisfying the required conditions. ■

Clearly lifted SAT includes SAT as a special case, and hence is NP-hard. The above theorem shows that the problem is in NP because we can nondeterministically guess the equivalence relation and truth assighment.

We will now reduce planning to lifted SAT, and then reduce lifted SAT to SAT. The composition of these two reductions will yield the desired concise reduction from planning to SAT. In the reduction from planning to lifted SAT we create a "fresh copy" of each operator definition at each step. For each step name o we let A_o consist of a fresh copy of each operator definition. Fresh copies are built by renaming the formal parameters. The variables appearing in A_o are disjoint from the variables appearing in A_p for distinct o and p. In the blocks world we have that A_o contains only a single element of the form $\texttt{MOVE}(x', y', z')$ where x', y' and z' are fresh variables for the step o. For any fixed operator set we have that $|A_o|$ is $O(1)$. We let A be the union of all A_o. Note that $|A|$ is $O(n)$. This should be contrasted to the ground case where $|A|$ is $O(|\text{Dom}|^{A_{\text{Ops}}})$. This reduction in $|A|$ is the main benefit of lifting. For any set of actions S we let $\text{Pre}(S)$ be the set of all prerequisites of

operations in S and similarly for Add(S) and Del(S). Note that $|\text{Pre}(A_o)|$ is $O(1)$, as are $|\text{Add}(A_o)|$ and $|\text{Del}(A_o)|$. Also note that Pre(A), Add(A), and Del(A) all have size $O(n)$. For convenience we also define A_I and A_F to be singleton sets of operators where A_I adds the initial assertions and A_F requires the final assertions. We will also use Vars(A) to denote the set of all variables appearing in A. Note that $|\text{Vars}(A)|$ is $O(n)$. The reduction from planning to lifted SAT can now be defined as in Table 2.

Selecting a way of satisfying schemas 1 and 2 selects the set of actions without making any commitment to the order of those actions. In our fomulation of the blocks world we have that A_o is a singleton set containing an action of the form $\text{MOVE}(x, y, z)$. Assuming that we are looking for a three step plan we have that O contains three step names and the first schema generates the following unit clauses.

$$\text{ACTION}(o_1, \text{MOVE}(x_1, y_1, z_1))$$

$$\text{ACTION}(o_2, \text{MOVE}(x_2, y_2, z_2))$$

$$\text{ACTION}(o_3, \text{MOVE}(x_3, y_3, z_3))$$

We now have nine variables. For an n step blocks world plan we will have $3n$ variables (independent of domain size). For any fixed set of operator definitions the total size of the instances of schemas 1 and 2 is $O(n)$.

Selecting a way of satisfying schema 3 selects a value for each variable. For each of the nine variables in the Sussman anomaly we have an instance of this schema. For example we have the following.

$$x_1 = \text{A} \vee x_1 = \text{B} \vee x_1 = \text{C} \vee$$
$$x_1 = \text{PLACE1} \vee x_1 = \text{PLACE2} \vee x_1 = \text{PLACE3}$$

For any fixed set of operator definitions the total size of instances of schema 3 is $O(n|\text{Dom}|)$.

We can think of the schemas as "running" nondeterministically from the top to the bottom. Schemas 1 and 2 select the actions and schema 3 selects the argument values. Schemas 4 and 5 assert the preconditions of each action. This is straightforward and the total size of the instances of these schemas is $O(n + |S_G|)$. Schema 6 selects a causal source for every precondition, i.e., it selects the set of causal links in the plan. Every causal link atomic formula of the form $o \xrightarrow{\Phi} p$ has the property that $\Phi \in \text{Pre}(A_p)$. Since $|\text{Pre}(A_p)|$ is $O(1)$ for $p \neq F$ and $O(|S_G|)$ for $p = F$, we have that the total number of causal link formulas is $O(n^2 + n|S_G|)$. The total size of the instances of schema 6 is $O(n^2 + n|S_G|)$.

Schema 7 states that the source of each link must precede its destination. The total size of this schema has the same order as schema 6. Schema 8 places a "nonlocal addition demand" on the source of each causal link. Since there is one such demand for each causal link, the total size of this schema is again $O(n^2 + n|S_G|)$. Note that the number "nonlocal" formulas of the form $\text{ADDS}(o, \Phi)$ is quadratic ($O(n^2 + n|S_G|)$). This should be contrasted with the linear number of "local" atomic formulas of the form $\text{NEEDS}(o, \Phi)$. In the Sussman anomaly we get formulas such as $\text{ADDS}(o_1, \text{CLEAR}(x_3))$.

After schema 8 forces certain addition formulas to be true, schemas 9, 10, and 11 select equations between the things to be added and the approriate elements of the add lists. The truth of these equations is "checked" by the semantics of lifted SAT. For example, in the Sussman anomaly we have (essentially) the following.

$$\text{ADDS}(o_1, \text{CLEAR}(x_3)) \Rightarrow \text{CLEAR}(x_3) = \text{CLEAR}(y_1)$$

The equation $\text{CLEAR}(x_3) = \text{CLEAR}(y_1)$ is equivalent to $x_3 = y_1$ and this equivalence is handled by the semantics of lifted SAT. The total size of the instances of schemas 9, 10, and 11 is $O(n^2 + n|S_0|)$.

Schema 12 forces nonlocal delete statements to be true. In the Sussman anomaly we have, essentially, the following.

$$\text{CLEAR}(x_3) = \text{CLEAR}(x_2) \Rightarrow \text{DELS}(o_3, \text{CLEAR}(x_2))$$

Note that there is a quadratic number of "nonlocal" deletion atomic formulas of the form $\text{DELS}(o, \Phi)$. Again this should contrasted with the linear number of "local" prerequisite assertions of the form $\text{NEEDS}(o, \Phi)$. The total size of the instances of schema 12 is $O(n^2 + n|S_G|)$.

Schema 13 handles the frame axioms by ensuring that deletions do not interfere with causal links. Schema 13 in the lifted encoding is analogous to schema 10 in the ground encoding. Schema 10 in the ground encoding generates $\Omega(n^3|\text{Dom}|^{A_{\text{Pred}}})$ clauses. However, schema 13 in the lifted version generates only $O(n^3 + n^2|S_G|)$ clauses. The key observation is that in the lifted encoding there are only $O(n^2 + n|S_G|)$ causal link formulas.

Schemas 14 and 15 check that the order conditions selected in schema 13 are consistent, i.e., that I comes first, F comes last, and that the transitive closure does not contain loops. The instances of these schemas involve $O(n^2)$ atomic formulas and order $O(n^3)$ clauses.

Overall we have a quadratic number of atomic formulas and a cubic number of literal occurrences. The number of atomic formulas is dominated by formulas of the form $x = c$, $\Phi = \Psi$, and causal links of the form $o \xrightarrow{\Phi} p$. All schemas have linear or quadratic size except for schemas 13 and 14 which are cubic.

To complete the reduction to SAT we give a reduction from general lifted SAT to SAT. Let $\mathcal{L}$ be a lifted SAT problem. In our reduction we simply add clauses to $\mathcal{L}$. The additional clauses ensure that equations occuring in $\mathcal{L}$ have the proper truth value and that two atomic formulas which are equal have the same truth value. We let $T(\mathcal{L})$ be the set of terms in $\mathcal{L}$ and we let $F(\mathcal{L})$ be the set of all atomic formulas in $\mathcal{L}$ other than equations. If $\mathcal{L}$ is the lifted SAT encoding of a planning problem then (for a fixed set of operator definitions) $|T(\mathcal{L})|$ is $O(|\text{Dom}| + n + |S_0| + |S_G|)$. In other words,

1.	$\texttt{ACTION}(o, a_1) \vee \ldots \vee \texttt{ACTION}(o, a_m)$	$o \in O,\ a_i \in A_o$
2.	$\neg(\texttt{ACTION}(o, a) \wedge \texttt{ACTION}(o, b))$	$o \in O,\ a, b \in A_o,\ a \neq b$
3.	$x = c_1 \vee \ldots \vee x = c_n$	$x \in \text{Vars}(A),\ c_i \in \text{Dom}$
4.	$\texttt{ACTION}(o, a) \Rightarrow \texttt{NEEDS}(o, \Phi)$	$o \in O,\ a \in A_o,\ \Phi \in \text{Pre}(a_i)$
5.	$\texttt{NEEDS}(F, \Phi)$	$\Phi \in \text{Pre}(F)$
6.	$\texttt{NEEDS}(p, \Phi) \Rightarrow (o_1 \xrightarrow{\Phi} p \vee \ldots \vee o_n \xrightarrow{\Phi} p)$	$p \in O \cup \{F\},\ \Phi \in \text{Pre}(A_p),\ o_i \in O \cup \{I\}$
7.	$o \xrightarrow{\Psi} p \Rightarrow o < p$	$o \in O \cup \{I\},\ p \in O \cup \{F\},\ \Psi \in \text{Pre}(A_p)$
8.	$o \xrightarrow{\Phi} p \Rightarrow \texttt{ADDS}(o, \Phi)$	$o \in O \cup \{I\},\ p \in O \cup \{F\},\ \Phi \in \text{Pre}(p)$
9.	$\texttt{ACTION}(o, a) \wedge \texttt{ADDS}(o, \Psi) \Rightarrow (\Psi = \Phi_1 \vee \ldots \vee \Psi = \Phi_n))$	$o \in O,\ a \in A_o,\ \Psi \in \text{Pre}(A \cup \{F\}),\ \Phi_j \in \text{Add}(a)$
10.	$\texttt{ADDS}(I, \Psi) \Rightarrow (\Psi = \Phi_1 \vee \ldots \vee \Psi = \Phi_n)$	$\Psi \in \text{Pre}(A),\ \Phi_i \in \text{Add}(I)$
11.	$\neg\texttt{ADDS}(I, \Psi)$	$\Psi \in \text{Pre}(F) - \text{Add}(I)$
12.	$\texttt{ACTION}(o, a) \wedge \Psi = \Phi \Rightarrow \texttt{DELS}(o, \Psi)$	$o \in O,\ a \in A_o,\ \Psi \in \text{Pre}(A \cup \{F\}),\ \Phi \in \text{Del}(a)$
13.	$o \xrightarrow{\Psi} p \wedge \texttt{DELS}(r, \Psi) \Rightarrow (r < o \vee p < r)$	$o \in O \cup \{I\},\ p \in O \cup \{F\},\ r \in O - \{o, p\}, \Psi \in \text{Pre}(A_p)$
14.	$o < r \wedge r < p \Rightarrow o < p$	$o, r, p \in O \cup \{I, F\}$
15.	$\neg(p < p)$	$p \in O \cup \{I, F\}$

Table 2: Lifted reduction of ground causal nonlinear planning.

we have only a linear number of terms. Furthermore, most atomic formulas in the lifted SAT encoding of a planning problem involve step name constants. This observation can be used to show that for planning problems there are only a quadratic number of unifiable pairs of formulas in $F(\mathcal{L})$. For any lifted SAT problem $\mathcal{L}$ we define the augmented clause set $\mathcal{L}'$ to be $\mathcal{L}$ plus the clauses in Table 3.

Schemas 1 through 5 ensure that the true equations define an equivalence relation on terms consistent with the interpretation of each function as a Herbrand constructor. Two terms "clash" if they are either distinct constants, one is a constant and one is a function application, or they are applications of different function symbols. Schema 5 states that clashing terms must not be equal.

Schemas 1 through 5 introduce $|T(\mathcal{L})|^2$ new atomic formulas — the equations of the form $t = u$. The number of instances of schema 4 can be no larger than $|T(\mathcal{L})|^2$. If we bound the arity (number of arguments) of functions then the total size of literal occurances in instances of schema 4 is $O(|T(\mathcal{L})|^2)$. For bounded arity, the the number of literal occurances in instances of schemas 1 through 5 is dominated by the transitivity schema, schema 3. The number of instances of schema 3 is $|T(\mathcal{L})|^3$.

Schemas 6, 7, and 8 handle the occurs-check condition. Without these schemas $x = f(y)$ and $y = f(x)$ appear consistent. But these equations (interpreted over the Herbrand universe of first order terms) imply that x is a subterm of itself. Since the Herbrand universe does not include infinite terms, this is impossible. These equations are inconsistent with schemas 6,7, and 8 because we can now infer $\texttt{OCCURS-IN}(x, x)$. Assuming a bounded arity for function symbols we have that schemas 6, 7, and 8 involve $O(|T(\mathcal{L})|^2)$ atomic formulas and $O(|T(\mathcal{L})|^3)$ clauses (and literal occurances).

Schema 7 enforces the condition that equivalent atomic literals have the same truth value. The number of instances of schema 7 equals the number of unifiable pairs of atomic formulas. We now have the following theorem.

Theorem: If $\mathcal{L}$ is a lifted SAT problem then $\mathcal{L}'$ as defined by the above augmentation is a satisfiable as a SAT problem if and only if $\mathcal{L}$ is satisfiable as a lifted problem. Furthermore, the number of additional atomic formulas in $\mathcal{L}'$ is $O(|T(\mathcal{L})|^2 + |U(\mathcal{L})|)$ where $U(\mathcal{L})$ is the number of unifiable pairs of atomic formulas. For bounded arity functions and predicates the number of additional literal occurances is $O(|T(\mathcal{L})|^3 + |U(\mathcal{L})|)$.

It is interesting to note that the above schemas can be converted to Horn clauses in linear time. When combined with a linear time algorithm for Boolean Horn clauses, the schemas define a cubic time unification algorithm. The classical Robinson unification algorithm is exponential time but a linear time algorithm is known (Patterson78).

It is also worth noting that in the lifted SAT problems that result from encodings of planning problems we need only schemas 1 through 5. In these lifted problems variables can only be bound to constants so no occurs-check reasoning is needed. Furthermore, all atomic formulas other than equations are forced to have appropriate truth values even in the absense of schema 7.

The total reduction from planning to SAT yields a quadratic number of atomic formulas and a cubic number of literal occurances.

6 CONCLUSIONS

Kautz and Selman (1996) challenged the widespread belief in the AI community that planning is not amenable to gen-

1.	$t = t$	$t \in T(\mathcal{L})$
2.	$t = u \Rightarrow u = t$	$t, u \in T(\mathcal{L})$
3.	$t = u \wedge u = w \Rightarrow t = w$	$t, u, w \in T(\mathcal{L})$
4.	$(f(t_1, \ldots, t_n) = f(u_1, \ldots, u_n))$ $\equiv (t_1 = u_1 \wedge \ldots \wedge t_n = u_n)$	$f(t_1, \ldots, t_n) \in T(\mathcal{L})$ $f(u_1, \ldots, u_n) \in T(\mathcal{L})$
5.	$\neg(t = u)$	$t, u \in T(\mathcal{L})$, t, u clash
6.	OCCURS-IN$(t_i, f(t_1, \ldots, t_n))$	$f(t_1, \ldots, t_n) \in T(\mathcal{L})$
7.	OCCURS-IN$(u, w) \wedge$ OCCURS-IN(w, t) $\Rightarrow$ OCCURS-IN(u, t)	$u, w, t \in T(\mathcal{L})$
8.	$\neg$OCCURS-IN(t, t)	$t \in T(\mathcal{L})$
9.	$P(t_1, \ldots, t_n) \wedge t_1 = u_1 \wedge \ldots \wedge t_n = u_n$ $\Rightarrow P(u_1, \ldots, u_n)$	$P(t_1, \ldots, t_n) \in F(\mathcal{L})$ $P(u_1, \ldots, u_n) \in F(\mathcal{L})$ $\forall i$ t_i, u_i unifiable

Table 3: Additional clauses for reduction from lifted SAT to ordinary SAT.

eral theorem-proving techniques, by showing that general propositional satisfiability algorithms can outperform specialized planning systems on a range of benchmark problems. Critical to the success of this approach is the use of concise SAT encodings of the planning problems. This paper described general, polynomial-time reductions from STRIPS-style planning to CNF formulas. We compared the various encodings, both in terms of the number of variables used and the total size of the formulas. This kind of analysis will allow one to use the gross statistical properties of a given planning problem to select a practical encoding.

This paper also introduced lifted causal encodings, a new kind of reduction. We showed that this encoding strictly dominates others in asymptotic terms. In future work, we will empirically evaluate this new class of encodings, both to determine whether the constant factors in the size of the encoding are reasonable for practical problems, and to determine whether our satisfiability procedures also perform well on this class of formulas.

References

Bacchus, F. and Kabanza, F. (1995). Using temporal logic to control search in a forward chaining planner. *Proc. EWSP-95*, 157–169.

Backstrom, C. (1992). *Computational complexity of reasoning about plans*, Ph.D. thesis, Linkoping University, Linkoping, Sweden.

Barrett, A. and Weld, D. (1994). Partial-order planning: evaluating possible efficiency gains. *Artificial Intelligence*, 67:71-112, 1994.

Bedrax-Weiss, T., Jonsson, A.K., and Ginsberg, M.L. (1996). Partial-order planning: evaluating possible efficiency gains. Unpublished manuscript.

Blum, A. and Furst, M.L. (1995). Fast planning through planning graph analysis. *Proc. IJCAI-95*, Montreal, Canada.

Bylander, T. (1991). Complexity results for planning. *Proc. IJCAI-91*, Sidney, Australia, 274-279.

Crawford, J.M. and Auton, L.D. (1993) Experimental Results on the Cross-Over Point in Satisfiability Problems. *Proc. AAAI-93*, Washington, DC, 21–27.

Davis, M., Logemann, G., and Loveland, D. (1962). A machine program for theorem proving. *Comm. ACM*, 5, 1962, 394–397.

Erol, K., Nau, D.S., and Subrahmanian, V.S. (1992). On the complexity of domain-independent planning. *Proc. AAAI-92*, 381–386.

Fikes, R.E. and Nilsson, N.J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3/4), 189–208.

Gupta and Nau (1992). On the complexity of blocks-world planning. *Artificial Intelligence*, **56**, 139–403.

Haas, A. (1987). The case for domain-specific frame axioms. *The Frame Problem in Artificial Intelligence, Proceedings of the 1987 Workshop*, F.M. Brown, ed., Lawrence, KS, 1987. Morgan Kaufmann Publishers, Los Altos, CA.

Joslin, D. and Pollack, M. (1995). Passive and Active Decision Postponement in Plan Generation. *European Workshop on Planning (EWSP)*, Assisi, Italy, Sept. 1995.

Kambhampati, S. and Yang, X. (1996). On the role of Disjunctive Representations and Constraint Propagation in Refinement Planning. *Proc. KR-96*.

Kautz, H. , McAllester, D., and Selman, B. (1996). Planning in Propositional Logic. In preparation.

Kautz, H. and Selman, B. (1992) Planning as Satisfiability. *Proc. ECAI-92*, Vienna, Austria, 1992, 359–363.

Kautz, H., and Selman, B. (1996). Pushing the envelope: planning, propositional logic, and stochastic search. *Proc. AAAI-96*, Portland, OR, 1996.

McCarthy, J. and Hayes, P. (1969). Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence 4*, D. Michie, ed., Ellis Horwood, Chichester, England, 1969, page 463ff.

McAllester, D. and Rosenblitt, D. (1991). Systematic nonlinear planning. *Proc. AAAI-91*, Anaheim, CA.

Mitchell, D., Selman, B., and Levesque, H.J. (1992). Hard and Easy Distributions of SAT Problems. *Proc. AAAI-*

92, San Jose, CA, 459–465.

Patterson, M.S. and Wegman, M.N. (1978). Linear unification. *JCSS*, 16 1978, 158–167.

Penberthy, J. and Weld, D. (1992). UCPOP: A sound, complete, partial order planner for ADL. In the *Proc. KR-92*, Boston, MA, 103–114.

Schubert, L. (1989). Monotonic Solution of the Frame Problem in the Situation Calculus: an Efficient Method for Worlds with Fully Specified Actions. *Knowledge Representation and Defeasible Reasoning*, H. Kyburg, R. Loui, and G. Carlson, eds.

Selman, B. (1994). Near-Optimal Plans, Tractability, and Reactivity. *Proc. KR-94*, Bonn, Germany, 1994, 521–529.

Selman, B., Kautz, H., and Cohen, B. (1996) Local Search Strategies for Satisfiability Testing. *Dimacs Series in Discrete Mathematics and Theoretical Computer Science*. (to appear)

Selman, B., Levesque, H., and Mitchell, D. (1992). A New Method For Solving Hard Satisfiability Problems. *Proc. AAAI-92*, San Jose, CA, 1992, 440-446.

Slaney, J. and Thiebaux, S. (1996). Linear Time Near-Optimal Planning in the Blocks World. *Proc. AAAI-96*.

Stone, P., Veloso, V., and Blythe, J. (1994). The need for different domain-independent heuristics. In *AIPS94*, pages 164-169, Chicago, 1994.

Veloso, M. (1992). Learning by analogical reasoning in general problem solving. Ph.D. Thesis, CMU, CS Techn. Report CMU-CS-92-174.

Deductive Systems

Implementing Modal and Relevance Logics in a Logical Framework

David Basin, Seán Matthews, Luca Viganò
Max-Planck-Institut für Informatik
Im Stadtwald, D-66123 Saarbrücken, Germany
{basin,sean,luca}@mpi-sb.mpg.de

Abstract

We present a framework for machine implementation of both partial and complete fragments of large families of non-classical logics such as modal, relevance, and intuitionistic logics. We decompose a logic into two interacting parts, each a natural deduction system: a base logic of labelled formulae, and a theory of labels characterizing the properties of the Kripke models. Our approach is modular and supports uniform proofs of correctness and proof normalization. We have implemented our work in the Isabelle Logical Framework.

1 INTRODUCTION

The origins of natural deduction (ND) are both philosophical and practical. In philosophy, it arises from an analysis of deductive inference in an attempt to provide a theory of meaning for the logical connectives [24, 33]. Practically, it provides a language for building proofs, which can be seen as providing the deduction theorem directly, rather than as a derived result. Our interest is on this practical side, and a development of our work on applications of *logical frameworks*, i.e. formal notations providing support for the uniform implementation of different logics, based on fragments of higher-order implicational logic and suitable in particular for ND [15, 22, 30].

We address the problem of how to present families of related non-classical logics so as to be suitable for such implementations. The problem is not trivial: these logics are usually presented as Hilbert systems and, even if a presentation is an ND system, it is often specialized, and metatheorems such as soundness and completeness (with respect to the semantics) sometimes have significantly different proofs, even for closely related logics. As a result, finding 'good' presentations is still specialist work.[1]

The particular *non-classical* logics we consider are those with *non-classical* connectives (we also use *modality* and *non-local connective* as synonyms for *non-classical connective*) which can, e.g., assert the necessity or contingency of propositions, or take account, in some way, of context. Many of these logics can be interpreted using a Kripke semantics of 'worlds' and relations between them, where the meaning of a non-classical connective at some world is defined in terms of conditions at others; e.g. the $\Box$ and $\Diamond$ of modal logic, interpreted using binary relations [16], or relevant implication, using a ternary relation [6], or non-classical negation, again using a binary relation [5, 7]. In each case a class of logics is defined by variations of the behavior of the relation. Using this view of non-classical logics to build ND presentations, we are able to (i) exploit modularity in the semantics so that related logics result from modifications just to the behavior of the relations, and (ii) provide basic metatheoretic results in a modular fashion; e.g. the soundness and completeness of encodings, and proof normalization results, are parameterized over the properties of the relations.

ND, even though recognized as one of the more practical notations for a proof system (cf. [32]), is usually seen as ill-suited for non-classical logics, because it builds in too many assumptions about the logic it is encoding. Proof under assumption needs a deduction theorem: 'if assuming A true, we can show B true, then $A \rightarrow B$ is true'; but for implications weaker or substantially different from intuitionistic $\rightarrow$ this fails (at least for the conventional reading of 'if-then'). At-

[1] With sufficient effort, a logical framework can implement *any* (recursively enumerable) proof system, but the resulting encoding does not necessarily 'fit' well (see [13], where a concept of a *natural representation* in a framework is formalized and investigated).

tempts to build ND presentations of non-classical logics have thus introduced various technical devices to get around these problems. For example, Dunn [6], for relevance logics, considers 'relevant' ND, where rules have side conditions on discharged assumptions, and Prawitz [24], for (some) modal logics, proposes rules for $\Box$ with side conditions that the main connective of all the supporting assumptions is $\Box$. The continuing primacy of Hilbert presentations in non-classical logic, despite the difficulty in actually using them, is evidence that these devices have not been completely successful.

Nevertheless, in this paper we present non-classical logics as ND proof systems. Our systems however, unlike those of Dunn or Prawitz, fit well in a standard framework: all our rules are *ordinary* (insensitive to thinning or contraction of assumptions), *pure* (have no non-local side conditions), and *single-conclusioned*.[2] Our presentations are partitioned into interacting parts: a *base logic* of propositional[3] logic and a separate *relational theory* characterizing the properties of the relations; the base logic stays fixed for a given class of related logics and we generate the one we want by 'plugging in' the appropriate relational theory. Thus the task we address here is how to reduce non-classical logics to such interacting ND theories with well-behaved proof rules for which we have general metatheorems about correctness and other desirable properties. To carry out this program, we combine the language of ND with that of a *Labelled Deductive System* (LDS), as proposed by Gabbay [12] and others (e.g. [11, 31]). We show that for many non-classical logics with Kripke-style semantics, including modal and relevance logics, it is possible to use LDS to partition presentations and give well-behaved LDS/ND systems supporting the modularity we want.

To illustrate, take the example of modal logic, where the usual deduction theorem fails. The standard Kripke completeness theorem tells us that A is provable ($\vdash A$) iff A is true at every world in every suitable frame (W, R), where W is the set of possible worlds, and R is the accessibility relation between worlds. Then $\vdash A$ iff $\forall w \in \mathsf{W}(w \models A)$, and the deduction theorem, as formulated above, corresponds to

$$\forall w \in \mathsf{W}(w \models A) \Rightarrow \forall w \in \mathsf{W}(w \models B) \Rightarrow \forall w \in \mathsf{W}(w \models A \to B),$$

where $\Rightarrow$ is implication in the meta-language and $\to$ is implication in the object language. But this is false, since from the semantics we have only that

$$\forall w \in \mathsf{W}(w \models A \Rightarrow w \models B) \Rightarrow \forall w \in \mathsf{W}(w \models A \to B).$$

Thus a naïve embedding of a modal logic in an ND system is going to fail. Suppose, however, we extend ND to be over pairs drawn from the language of modal logic *and labels*; i.e. instead of $\vdash A$, we consider $\vdash w{:}A$, where w is a 'world', and $\forall w \in \mathsf{W}(\vdash w{:}A)$ iff $\vdash A$. This provides a language to formulate the above theorem, and provides a basis for an ND system. Moreover, we can use the same notation to express the general behavior of modal operators like $\Box$ in a way that is independent of the details of the Kripke model providing the semantics, i.e. $\vdash w{:}\Box A$ iff $\vdash w'{:}A$ for all $w' \in \mathsf{W}$ accessible from w. Then by formalizing the details of particular accessibility relations we can produce particular modal logics. This treatment has obvious similarities to traditional semantic embedding (i.e. translation into predicate logic, e.g. [19]), but it offers substantial advantages: our formalization does not require all of first-order logic and it yields structured ND systems where the strong separation between the base logic and the relational theory gives us better proof normalization results (cf. Theorem 11 and its commentary).

In [2] we investigated LDS/ND for modal logics based on classical propositional logic. This paper explores the generalizations needed to build proof systems that handle both positive and full fragments of large families of non-classical logics, including relevance and intuitionistic (and classical) logic, and can treat non-classical negation as a modality (the metatheory of positive logics is different from that for full logics, cf. Dunn's semantic treatment of positive modal logics in [9]). We provide a framework for a uniform treatment of a wide range of non-classical connectives ($\Box$, $\Diamond$, relevant implication, non-classical negation, etc.). Our framework is based on an abstract classification of modalities as 'universal' and 'existential', and on general metatheorems about them. We have implemented our approach in the Isabelle system, which supports management of separate theories and their structured combination, resulting in a parameterized proof development system where (although it is not formally quantifiable) proof construction is natural and intuitive.

Due to lack of space, proofs have been omitted or considerably shortened, see [3] for details.

[2] We use the vocabulary of [1], which notes (§5.5) that 'every ordinary, pure single-conclusioned ND system can, e.g., quite easily be implemented in the Edinburgh LF.'

[3] We do not, here, consider quantified non-classical logics, which can be presented by means of quantifier rules similar to those of free logic.

2 LABELLED NON-CLASSICAL LOGICS

In this section we formalize our presentations. We introduce the fundamentals of how an LDS presentation relates to a Kripke semantics (Section 2.1). After this we define the base logic (Section 2.2) and the associated class of relational theories over which it is parameterized (Section 2.3). Finally, we give examples of non-classical logics (Section 2.4).

2.1 LABELS AND KRIPKE MODELS

Let W be a set of *labels*, ranging over worlds in a Kripke model, and R an $n+1$-ary relation over W. If $a, a_1, \ldots, a_n$ are labels and A is a formula, then we call $Ra\,a_1 \ldots a_n$ a *relational formula* (*rwff*), and $a{:}A$ a *labelled formula* (*lwff*). We partition connectives in a logic into two families: *local* and *non-local*. If a formula A is built from a local connective $\mathcal{C}$ of arity n, $A = \mathcal{C}(A_1, \ldots, A_n)$, then the truth of the lwff $a{:}A$ depends only on the (local) truth of $a{:}A_1, \ldots, a{:}A_n$. Typical local connectives are conjunction ($\wedge$), disjunction ($\vee$), material implication ($\supset$), and 'local' negation ($\sim$). Where $\models^{\mathsf{M}}$ is the truth relation for lwffs in the model M, we have:[4]

$$\models^{\mathsf{M}} a{:}A \wedge B \text{ iff } \models^{\mathsf{M}} a{:}A \text{ and } \models^{\mathsf{M}} a{:}B \quad (1)$$

$$\models^{\mathsf{M}} a{:}A \vee B \text{ iff } \models^{\mathsf{M}} a{:}A \text{ or } \models^{\mathsf{M}} a{:}B \quad (2)$$

$$\models^{\mathsf{M}} a{:}A \supset B \text{ iff } \models^{\mathsf{M}} a{:}A \text{ implies } \models^{\mathsf{M}} a{:}B \quad (3)$$

$$\models^{\mathsf{M}} a{:}{\sim}A \text{ iff } \not\models^{\mathsf{M}} a{:}A \quad (4)$$

For notational simplicity, we omit parentheses where possible and write binary connectives in infix notation (as above).

A non-local connective $\mathcal{M}$ of arity n is associated with an $n+1$-ary relation R on worlds, and the truth of $a{:}\mathcal{M}A_1 \ldots A_n$ is evaluated non-locally at the worlds R-accessible from a; i.e. it depends on the truth of $a_1{:}A_1, \ldots, a_n{:}A_n$ where $Ra\,a_1 \ldots a_n$.

Examples of non-local connectives and associated relations are the unary modal operator $\Box$ and the binary accessibility relation on possible worlds, or relevant implication $\rightarrow$ and the ternary compossibility relation. We extend $\models^{\mathsf{M}}$ to express truths for rwffs in a Kripke model M with an $n+1$-ary relation R as

$$\models^{\mathsf{M}} Ra\,a_1 \ldots a_n \text{ iff } (a, a_1, \ldots, a_n) \in \mathsf{R}, \quad (5)$$

and we call $\mathcal{M}$ a *universal non-local connective* when the metalevel quantification in the evaluation clause of $\mathcal{M}$ is universal (and the body is an implication):

$$\models^{\mathsf{M}} a{:}\mathcal{M}A_1 \ldots A_n \text{ iff for all } a_1, \ldots, a_n \\ ((\models^{\mathsf{M}} Ra\,a_1 \ldots a_n \text{ and } \models^{\mathsf{M}} a_1{:}A_1 \text{ and} \ldots \text{and} \\ \models^{\mathsf{M}} a_{n-1}{:}A_{n-1}) \text{ imply } \models^{\mathsf{M}} a_n{:}A_n). \quad (6)$$

Similarly, $\mathcal{M}$ is an *existential non-local connective* when the metalevel quantification is existential (and the body is a conjunction):

$$\models^{\mathsf{M}} a{:}\mathcal{M}A_1 \ldots A_n \text{ iff there exist } a_1, \ldots, a_n \\ (\models^{\mathsf{M}} Ra\,a_1 \ldots a_n \text{ and } \models^{\mathsf{M}} a_1{:}A_1 \text{ and} \ldots \text{and} \\ \models^{\mathsf{M}} a_{n-1}{:}A_{n-1} \text{ and } \models^{\mathsf{M}} a_n{:}A_n). \quad (7)$$

Thus $\Box$ and (relevant) $\rightarrow$ are universal non-local connectives, $\Diamond$ is existential, and their customary evaluation clauses are special cases of (6) and (7), e.g.

$$\models^{\mathsf{M}} a{:}A_1 \rightarrow A_2 \text{ iff for all } a_1, a_2((\models^{\mathsf{M}} Ra\,a_1\,a_2 \\ \text{and } \models^{\mathsf{M}} a_1{:}A_1) \text{ imply } \models^{\mathsf{M}} a_2{:}A_2). \quad (8)$$

A uniform treatment of negation plays a central role in our framework. However, in the Kripke semantics for relevance (and other) logics, it may be necessary for both a formula and its negation to be true at a world, which cannot be the case with $\sim$; thus a new connective is introduced, a *non-local negation* $\neg$, formalized by a unary function $*$ on worlds [6]:

$$a \models^{\mathsf{M}} \neg A \text{ iff } a^* \not\models^{\mathsf{M}} A. \quad (9)$$

Informally, a^* is the world which does not deny what a asserts, i.e. a and a^* are compatible worlds. We generalize this by introducing the constant $\perp\!\!\!\perp$ that expresses *incoherence* of compatible worlds to replace (9) with

$$\models^{\mathsf{M}} a{:}\neg A \text{ iff for all } b \\ (\models^{\mathsf{M}} a^*{:}A \text{ implies } \models^{\mathsf{M}} b{:}\perp\!\!\!\perp), \quad (10)$$

where $\not\models^{\mathsf{M}} b{:}\perp\!\!\!\perp$ for every world b.

Some remarks. First, when relevant implication is present, we can define $a{:}\neg A$ as $a{:}A \rightarrow \perp\!\!\!\perp$, and postulate $Ra\,a^*\,b$ for every b, so that (10) is just a special case of (8).[5] Second, when $a = a^*$, e.g. for modal or classical logic, $\perp\!\!\!\perp$ reduces to $\perp$, $\neg$ to $\sim$, and (10) to

[4] $\sim$ can also be defined in terms of $\supset$ and *falsum* ($\perp$). Then we can compare, like for modal logics (cf. [2]), the logics obtained when (i) $\models^{\mathsf{M}} a{:}{\sim}A$ iff $\models^{\mathsf{M}} a{:}A$ implies $\models^{\mathsf{M}} b{:}\perp$, and (ii) $\perp$ is 'global', i.e. $\models^{\mathsf{M}} a{:}\perp$ implies $\models^{\mathsf{M}} b{:}A$, with the (paraconsistent) logics where (i') $\models^{\mathsf{M}} a{:}{\sim}A$ iff $\models^{\mathsf{M}} a{:}A$ implies $\models^{\mathsf{M}} a{:}\perp$, and (ii') $\perp$ is 'local', i.e. $\models^{\mathsf{M}} a{:}\perp$ implies $\models^{\mathsf{M}} a{:}A$.

[5] That a and a^* are 'compossible' according to every world b is justified by the meaning of $*$.

(4). Finally, there is a well-known approach to non-local negation, e.g. for relevance, linear and ortho-logic (cf. [5, 7, 14, 27]), which uses an *incompatibility relation* N between worlds:

$$\models^{\mathsf{M}} a{:}\neg A \text{ iff for all } b(\models^{\mathsf{M}} b{:}A \text{ implies } b\ N\ a). \quad (11)$$

Then a^* is the 'strongest' world b for which $b\ N\ a$ does *not* hold. This can be shown equivalent to our approach (for a comparison of (11) with (9) see [7]).

Hence, we define the language of a non-classical logic L as follows:

Definition 1 Let W be a set of labels, and I, J two finite sets of indices. The language of a non-classical logic L is a tuple (W, S, O, F). S is a denumerably infinite set of sentence letters. O is the set whose members are (i) the constant $\perp\!\!\!\perp$ (and/or $\perp$); (ii) local and/or non-local negation (or neither for positive logics); (iii) a set of local connectives $\{C_j \mid j \in J\}$; (iv) a set of non-local connectives $\{\mathcal{M}_i \mid i \in I\}$ with an associated set $\vec{R} = \{R_i \mid i \in I\}$ of relations of the appropriate arities. F is the set of rwffs and lwffs: if $a, a_1, \ldots, a_n$ are labels, R_i has arity $n+1$, and A is a formula built up from members of S and O, then $R_i\, a\, a_1 \ldots a_n$ is an rwff and $a{:}A$ is an lwff.

Note that by associating different relations to universal and existential non-local connectives, we make no a priori assumptions about their interrelationships (when the relations are not independent, incompleteness may arise, cf. Theorem 10 in Section 3.1).

Definition 2 A *frame* (for the logic L) is a tuple $(\mathsf{W}, \mathsf{G}, \vec{\mathsf{R}}, *)$, where W is a non-empty set of *worlds*, $\mathsf{G} \in \mathsf{W}$ is the *actual world*, $\vec{\mathsf{R}} = \{\mathsf{R}_i \mid i \in I\}$ is the set of relations over W corresponding to $\vec{R}$, and $*$ is a unary function on worlds. A *model* M (for L) consists of a frame and a function V mapping elements of W and sentence letters to truth values (0 or 1), where

$$\models^{\mathsf{M}} a{:}p \text{ iff } \mathsf{V}(a, p) = 1 \quad (12)$$

and $\models^{\mathsf{M}}$ is extended to lwffs with local and non-local connectives and to rwffs as above. When $\models^{\mathsf{M}} \varphi$, for φ an lwff or an rwff, we say that φ is *true* in M.

A non-classical logic L is then characterized by its language and by its models, e.g. the conditions independently imposed on $*$ and each R_i. Moreover, some logics, e.g. intuitionistic and relevance (but not classical modal) logics, require truth to be monotonic. We define a partial order $\sqsubseteq$ on worlds, where, e.g., for intuitionistic logic $\sqsubseteq$ coincides with the accessibility relation, while for relevance logics it can be defined in terms of R, i.e. $a \sqsubseteq b$ iff $RGab$. For modal logic $\sqsubseteq$ reduces to equality. Then we require that V satisfy the *atomic monotony condition*, i.e. for any a_i, a_j and for any sentence letter p:

$$\text{if } \models^{\mathsf{M}} a_i{:}p \text{ and } \models^{\mathsf{M}} a_i \sqsubseteq a_j, \text{ then } \models^{\mathsf{M}} a_j{:}p. \quad (13)$$

One might be tempted to generalize this immediately to arbitrary formulae; this is in fact the case for 'usual' non-classical logics, such as intuitionistic and relevance logics, where we can prove by induction on the structure of A that

$$\text{if } \models^{\mathsf{M}} a_i{:}A \text{ and } \models^{\mathsf{M}} a_i \sqsubseteq a_j, \text{ then } \models^{\mathsf{M}} a_j{:}A. \quad (14)$$

But there are logics for which (14) does not hold for every formula; e.g. [10, 17] combine intuitionistic implication $\rightarrow$ with classical implication $\supset$, and show that (14) holds for $A \rightarrow B$ (in fact it holds, as one would expect, for every intuitionistic formula) but it fails for $A \supset B$. This problem is solved in [10, 17] by restricting (14) to *persistent* formulae: A is persistent if (i) it is atomic, or (ii) if it is of the form $B \rightarrow C$ or $\neg B$, where $\neg$ is intuitionistic (and thus non-local) negation, or (iii) it is of the form $B \wedge C$ or $B \vee C$, and B and C are both persistent. Similar definitions can be given for other non-classical logics, depending on the particular language we are considering. With such a precaution (14) refines to the following property (provable from (13) by induction on the structure of A):

Property 3 For any a_i, a_j and for any persistent formula A, if $\models^{\mathsf{M}} a_i{:}A$ and $\models^{\mathsf{M}} a_i \sqsubseteq a_j$, then $\models^{\mathsf{M}} a_j{:}A$.

Monotony is defined also for rwffs: for an $n+1$-ary relation R_i we require that if $\models^{\mathsf{M}} R_i\, a_0 \ldots a_j \ldots a_n$ then

$$\text{if } \models^{\mathsf{M}} a_i \sqsubseteq a_j, \text{ then } \models^{\mathsf{M}} R_i\, a_0 \ldots a \ldots a_n, \quad \text{for all } j < n \quad (15)$$

$$\text{if } \models^{\mathsf{M}} a_n \sqsubseteq a, \text{ then } \models^{\mathsf{M}} R_i\, a_0 \ldots a_{n-1}\, a. \quad (16)$$

In the following we assume formulae of the form $a \sqsubseteq b$ to be special cases of relational formulae, but we note that one could introduce them explicitly as a third kind of formulae, independent of lwffs and rwffs (proof-theory and semantics are then extended accordingly). This assumption allows us to treat the properties of the partial order, reflexivity and transitivity, as instances of (15) and (16).

As a notational simplification, we will restrict our attention to non-classical logics with a restricted language containing the local connectives $\wedge, \vee, \supset$, one

universal non-local connective $\mathcal{M}^u$ of arity u associated with a relation R^u of arity $u+1$, one existential non-local connective $\mathcal{M}^e$ of arity e associated with a relation R^e of arity $e+1$, non-local negation $\neg$, and the constant $\perp\!\!\!\perp$.[6] Then, from Definition 2, a model for such a logic is the tuple $\mathsf{M} = (\mathsf{W}, \mathsf{G}, \mathsf{R}^u, \mathsf{R}^e, *, \mathsf{V})$, and truth for an rwff or lwff φ in M, $\models^{\mathsf{M}} \varphi$, is the smallest relation $\models^{\mathsf{M}}$ satisfying (1), (2), (3), (5) and (6) for R^u and $\mathcal{M}^u$, (5) and (7) for R^e and $\mathcal{M}^e$, (10), (12), (13), (15) and (16) for R^u and R^e.

Finally note that we do not, here, consider logics like the relevance logic **E** for which models with more than one actual world are needed. These logics can be formalized by considering a set P of actual worlds and modifying the postulates of the relational theory with a precondition testing membership in P: for instance, for identity the postulate $RGaa$ is replaced with '$x \in P$ implies $Rxaa$'.

2.2 THE BASE LOGIC $\mathcal{B}$

We now introduce the base logic $\mathcal{B}$ that provides the rules we need to reason about lwffs. Our formalization of the base logic is motivated by pragmatic concerns: (i) it should make no assumptions on the relational theories extending it, (ii) it should be adequate for the logics we are interested in, and (iii) it should have good proof-theoretic properties. In [2] we are able to provide a base logic for the modal logics of the Geach hierarchy that satisfies all these criteria. Unfortunately, in the more general case considered here, things are not so clear cut: to achieve (ii) and (iii) we have to replace (i) with (i') it should make as few assumptions as possible on the relational theories extending it and ideally none at all, depending on the logic we want to formalize (cf. Section 2.3, where we discuss 'complementary rules', and Section 4, where we discuss extensions with first-order relational theories).

We start by considering the simplest connective: classical (local) implication. For this we adapt the 'traditional' ND rules, simply adding a label, to get the rules $\supset I$ and $\supset E$:

$$\frac{\begin{array}{c}[a{:}A_1]\\ \vdots\\ a{:}A_2\end{array}}{a{:}A_1 \supset A_2}\supset I \qquad \frac{a{:}A_1 \supset A_2 \quad a{:}A_1}{a{:}A_2}\supset E \tag{17}$$

Rules for $\wedge$, $\vee$, or other local connectives, are adapted similarly. Then we give the rules for $\mathcal{M}^u$ and $\mathcal{M}^e$:

$$\frac{[\overrightarrow{a{:}A}^{u-1}]\,[R^u\,a\,a_1\ldots a_u]\quad\vdots\quad a_u{:}A_u}{a{:}\mathcal{M}^u A_1\ldots A_u}\,\mathcal{M}^uI$$

$$\frac{a{:}\mathcal{M}^u A_1\ldots A_u \quad \overrightarrow{a{:}A}^{u-1} \quad R^u\,a\,a_1\ldots a_u}{a_u{:}A_u}\,\mathcal{M}^uE$$

$$\frac{\overrightarrow{a{:}A}^{e} \quad R^e\,a\,a_1\ldots a_e}{a{:}\mathcal{M}^e A_1\ldots A_e}\,\mathcal{M}^eI$$

$$\frac{a{:}\mathcal{M}^e A_1\ldots A_e \qquad \begin{array}{c}[\overrightarrow{a{:}A}^{e}]\,[R^e\,a\,a_1\ldots a_e]\\ \vdots\\ b{:}B\end{array}}{b{:}B}\,\mathcal{M}^eE \tag{18}$$

where $\overrightarrow{a{:}A}^{n}$ stands for the finite sequence of premises $a_1{:}A_1\ldots a_n{:}A_n$. Moreover, in $\mathcal{M}^uI$ and $\mathcal{M}^eE$, each a_k, a_l $(1 \le k \le u-1,\ 1 \le l \le e)$ is fresh; e.g. in $\mathcal{M}^eE$, $a_1,\ldots,a_e$ are all different from a, b and each other, and do not occur free in $b{:}B$ or in any assumption other than those listed.

Comparing these rules with (3), (6) and (7), we see that they reflect the semantic definitions. When we treat negation, however, the correspondence between the rules and the semantics is more subtle, and we must choose which kind of negation we want to encode. We begin by providing rules for treating $\neg$ with the rules $\neg I$ and $\neg E$ (cf. (10)):

$$\frac{\begin{array}{c}[a^*{:}A]\\ \vdots\\ b{:}\perp\!\!\!\perp\end{array}}{a{:}\neg A}\,\neg I \qquad \frac{a{:}\neg A \quad a^*{:}A}{b{:}\perp\!\!\!\perp}\,\neg E \tag{19}$$

These rules capture only a *minimal* non-local negation, and if we want a base logic capable of formalizing *intuitionistic* or *classical* non-local negation we need the additional rules $\perp\!\!\!\perp Ei$ and $\perp\!\!\!\perp Ec$, respectively:

$$\frac{b{:}\perp\!\!\!\perp}{a{:}A}\,\perp\!\!\!\perp Ei \qquad \frac{\begin{array}{c}[a{:}\neg A]\\ \vdots\\ b{:}\perp\!\!\!\perp\end{array}}{a^*{:}A}\,\perp\!\!\!\perp Ec \tag{20}$$

Finally, we require the rule *monl*, expressing monotony at the level of lwffs:

$$\frac{a_i{:}A \quad a_i \sqsubseteq a_j}{a_j{:}A}\,monl \tag{21}$$

[6] Since the language might not contain (the non-local) $\rightarrow$, we take $\neg$ as a primitive operator as opposed to defined by means of $\perp\!\!\!\perp$ and $\rightarrow$.

where A is a persistent formula.[7] Since *monl* reflects Property 3, the details of its definition, including the proviso on its application, depend on the logic we are considering.

2.3 RELATIONAL THEORIES

We formalize a logic L by extending (the appropriate) $\mathcal{B}$ with a *relational theory* axiomatizing the properties of $*$ and of the relations R_i in a Kripke model. Correspondence theory [34, 35] and known correspondence results [29] allow us to determine which possible axioms correspond to which properties of R_i. Some of these properties can only be expressed using higher-order logic (e.g. the McKinsey axiom $\Box\Diamond A \supset \Diamond\Box A$), but for others first-order logic, or even fragments of it, is enough. We restrict our attention to properties axiomatizable using *(Horn) relational rules*, i.e. those of the form

$$\frac{R_i\, t_0^1 \ldots t_m^1 \quad \ldots \quad R_i\, t_0^n \ldots t_m^n}{R_i\, t_0 \ldots t_m}$$

where the t_k^j are terms built from labels and function symbols. (Some properties of R_i, e.g. *ass*1 and *ass*2 below, can be expressed as Horn relational rules only after the introduction of Skolem function constants, cf. [2].) A *(Horn) relational theory* $\mathcal{T}$ is then a theory generated by a set of such rules.

Even with such a restriction, we are able to capture many families of non-classical logics used in practice. For example, relational theories corresponding to logics in the modal Geach hierarchy (e.g. K, D, T, B, $S4$, $S4.2$, $KD45$, $S5$) [2], and various relevance logics (e.g. **B**, **N**, **T**, and **R**); cf. Section 2.4.[8]

Thus, for example, the modal axiom $\Box A \supset A$ corresponds to the reflexivity of the accessibility relation,

$$\frac{}{x\, R\, x}\ refl$$

while the $(A \to B) \to ((B \to C) \to (A \to C))$ and $A \to A$ axioms of relevance logic correspond to associativity and identity for the compossibility relation:

$$\frac{R a b e \quad R e c d}{R b\, h(a,b,c,d,e)\, d}\ ass1 \qquad \frac{R a b e \quad R e c d}{R a c\, h(a,b,c,d,e)}\ ass2$$

$$\frac{}{R G a a}\ iden \tag{22}$$

(Where h is a 5-ary Skolem function constant).

For negation we do not extend the properties of R_i directly, but instead refine the behavior of the $*$ function, which is part of the language of terms in the relational theory. Depending on whether we want to encode intuitionistic ($**i$), classical ($**i$, $**c$), or ortho ($o1$, $o2$) negation, we can select from the following rules, which impose different behaviors on $*$:

$$\frac{}{a \sqsubseteq a^{**}}\ **i \qquad \frac{}{a^{**} \sqsubseteq a}\ **c \qquad \frac{}{a \sqsubseteq a^{*}}\ o1 \qquad \frac{}{a^{*} \sqsubseteq a}\ o2$$

Finally, we have $n+1$ rules for the monotonic properties of rwffs (cf. (15), (16)):

$$\frac{R_i\, a_0 \ldots a_j \ldots a_n \quad a \sqsubseteq a_j}{R_i\, a_0 \ldots a \ldots a_n}\ monR_i(j), \text{ for } j < n$$

$$\frac{R_i\, a_0 \ldots a_n \quad a_n \sqsubseteq a}{R_i\, a_0 \ldots a_{n-1}\, a}\ monR_i(n)$$

Negation and monotony again raise the question of what exactly a base logic should be. The rules we have just given can be seen as rwff complements of lwff rules given earlier. For instance, for an intuitionistic negation, i.e. where the base logic contains $\perp\!\!\!\perp Ei$, we need $**i$, while for a classical negation, i.e. with $\perp\!\!\!\perp Ec$, we need $**i$ and $**c$; similarly, the $monR_i$ rules complement *monl*. Moreover, only by requiring these complementary rules can one establish desired proof-theoretic results (cf. the proof of Theorem 11 in [3]). Thus it is convenient, on pragmatic grounds, to assume that a base logic $\mathcal{B}$ is extended with a theory that includes these minimal relational rules (a characterization of the logics in which this complementarity is not satisfied, e.g. $\perp\!\!\!\perp Ei$ without $**i$, or $\perp\!\!\!\perp Ec$ with only $**c$, is out of the scope of this paper).

A logic $L = \mathcal{B} + \mathcal{T}$ is the extension of an appropriate base logic $\mathcal{B}$ with a Horn relational theory $\mathcal{T}$. Consider the restricted language (with $\wedge$, $\vee$, $\supset$, $\mathcal{M}^u$, $\mathcal{M}^e$, $\neg$, $\perp\!\!\!\perp$). Following Prawitz [24, 25], in Figure 1 we distinguish three families of ND systems according to their treatment of negation: minimal, intuitionistic or classical (we make the distinction by considering the $\perp\!\!\!\perp$ rules, as opposed to Prawitz's $\perp$ rules).

The *minimal system* $\mathcal{ML}$ is determined by a base logic containing *monl* (with the appropriate restrictions) and introduction and elimination rules for local (cf., e.g., (17)) and non-local connectives (cf. (18),

[7] If this restriction is not imposed, then the result is not sound for some logics, e.g. an attempted encoding of intuitionistic implication collapses to classical implication, similar to what is shown for Hilbert systems in [10, 17].

[8] Also, Horn relational rule sets can be directly implemented in the Horn fragment of the metalogics we use for our implementation (it is not necessary first to embed first-order logic or formalize additional judgements, cf. Section 5 and [15]). This restriction also yields good proof-theoretic properties (cf. Theorem 11).

L	$\mathcal{B}$	$\mathcal{T}$
$\mathcal{ML}$	rules for $\wedge, \vee, \supset, \mathcal{M}^u, \mathcal{M}^e, \neg$ *monl*	$monR_i$ rules
$\mathcal{JL}$	rules for $\wedge, \vee, \supset, \mathcal{M}^u, \mathcal{M}^e, \neg$ *monl* $\perp\!\!\!\perp Ei$	$monR_i$ rules $**i$
$\mathcal{CL}$	rules for $\wedge, \vee, \supset, \mathcal{M}^u, \mathcal{M}^e, \neg$ *monl* $\perp\!\!\!\perp Ec$	$monR_i$ rules $**i$, $**c$

Figure 1: The systems $\mathcal{ML}$, $\mathcal{JL}$ and $\mathcal{CL}$

and (19)), and by a relational theory containing, at least, the $monR_i$ rules, to complement *monl*.[9] The *intuitionistic system* $\mathcal{JL}$ is then obtained by extending $\mathcal{ML}$ with $\perp\!\!\!\perp Ei$ and the complementary rule $**i$, and the *classical system* $\mathcal{CL}$ is obtained by extending $\mathcal{ML}$ with $\perp\!\!\!\perp Ec$ and the complementary rules $**i$ and $**c$. For each of these systems we define:

Definition 4 If Γ is a set of lwffs, Δ a set of rwffs, and φ an lwff or an rwff, a *derivation* of φ from Γ and Δ in L is a tree formed using the rules in L, ending with φ and depending only on $\Gamma \cup \Delta$. We write $\Gamma, \Delta \vdash_L \varphi$ when φ can be so derived. A derivation of φ in L depending on the empty set, $\vdash_L \varphi$, is a *proof* of φ in L, and we say that φ is a *theorem* of L.

Fact 5 $\Gamma, \Delta \vdash_L R_i a a_1 \ldots a_n$ iff $\Delta \vdash_L R_i a a_1 \ldots a_n$.

2.4 EXAMPLES OF NON-CLASSICAL LOGICS

Our framework can be specialized to implement fragments and full presentations of large classes of modal and relevance logics. The important (though relatively simple) case of modal logics is discussed at length in [2]. There we show how the base logic K, which consists of the rules

$$\frac{\begin{array}{c}[x{:}A]\\ \vdots \\ x{:}B\end{array}}{x{:}A \supset B}\supset I \qquad \frac{x{:}A \supset B \quad x{:}A}{x{:}B}\supset E$$

$$\frac{\begin{array}{c}[x\,R\,y]\\ \vdots \\ y{:}A\end{array}}{x{:}\Box A}\Box I \qquad \frac{x{:}\Box A \quad x\,R\,y}{y{:}A}\Box E \qquad \frac{\begin{array}{c}[x{:}A \supset \bot]\\ \vdots \\ y{:}\bot\end{array}}{x{:}A}\bot E$$

[9] Note that, unlike Prawitz's, our minimal system does not satisfy the *inversion principle*, since it contains *monl* which is neither an introduction nor an elimination rule.

can be extended with relational rules to yield, e.g., logics in the Geach hierarchy. For instance, $S4$ is obtained by extending K with relational rules expressing reflexivity and transitivity of the accessibility relation:

$$\frac{}{x\,R\,x}\,refl \qquad \frac{x\,R\,y \quad y\,R\,z}{x\,R\,z}\,trans$$

Here we consider, as an example from relevance logics, the logic **R**: we compare our system with the Hilbert system $\mathbf{R_H}$ of Routley and Meyer [28], and show the advantages of our approach in the modular way we present the system so that it can be 'naturally' extended to obtain (positive and full) intuitionistic and classical logic.

Definition 6 We define **R** as follows. Since Routley and Meyer consider a classical non-local negation, (i.e. $\neg\neg A \rightarrow A$ is an axiom of $\mathbf{R_H}$), we use the classical version of $\mathcal{B}$ with $\perp\!\!\!\perp Ec$. $\mathcal{B}$ also includes *monl* and the rules for $\wedge$, $\vee$, $\neg$, and $\rightarrow$; $A \rightarrow B$ is defined as the binary universal modality $\mathcal{M}^u AB$ associated with the ternary relation R, for which we provide a relational theory generated by: $**i$ and $**c$, $monR(1)$ and[10] $monR(3)$, $ass1$, $ass2$, $iden$ (cf. (22)), and the further rules

$$\frac{}{Raaa}\,idem \qquad \frac{Rabc}{Rac^*b^*}\,anti \qquad \frac{Rabc}{Rbac}\,comm$$

$$\frac{Rabc}{Rabg(a,b,c)}\,cont1 \qquad \frac{Rabc}{Rg(a,b,c)bc}\,cont2$$

where g is a ternary Skolem function constant. $a \sqsubseteq b$ is defined to be $RGab$.

We get the positive fragment $\mathbf{R}^+$ simply by deleting all the rules involving non-local negation ($\neg I$, $\neg E$, $\perp\!\!\!\perp Ec$, $**i$, $**c$, *anti*).

We postpone proofs that **R** and $\mathbf{R}^+$ are what we claim they are, i.e. equivalent to $\mathbf{R_H}$ and $\mathbf{R_H}^+$ (which we get from $\mathbf{R_H}$ by deleting the axioms for negation), until Section 3, where we show the correctness of our presentations with respect to Kripke-style semantics. Here we are interested rather in comparing the modularity of the two presentations.

Routley and Meyer show that there is a problem with their presentations: $\mathbf{R_H}^+$ is a subsystem of positive intuitionistic logic, but $\mathbf{R_H}$ is a subsystem *only* of classical logic. That is, full intuitionistic logic **J** cannot be *modularly* obtained by simply adding new axioms to $\mathbf{R_H}$. (**J** can be obtained from $\mathbf{R_H}$, but only in a

[10] $monR(2)$ is derivable from $monR(1)$, since $Rabc \vdash_{\mathbf{R}} Rbac$.

non-modular fashion, if relevant negation is rejected in favour of an intuitionistic one [28, p.227].)

Now consider our systems. Positive intuitionistic logic $\mathbf{J}^+$ is obtained from $\mathbf{R}^+$ by adding the rule

$$\frac{}{RGGa}\ int$$

corresponding to the (intuitionistically valid 'thinning') axiom $A \to (B \to B)$, so that the ternary R reduces to a binary partial order (in fact to the usual accessibility relation for Kripke models of intuitionistic logic), and $\to$ reduces to intuitionistic implication. However, extending $\mathbf{R}$ with the rule *int* yields classical logic: we are able to derive $RGaG$, so that, essentially, all the worlds collapse; i.e. $a = a^* = a^{**}$, $\to$ reduces to $\supset$, and $\neg$ to $\sim$. This should not come as a surprise: in Definition 6 we explicitly defined $\mathbf{R}$ to contain, like $\mathbf{R_H}$, a classical treatment of negation. That is, with reference to Figure 1, we defined $\mathbf{R} = \mathbf{CR}$. But Figure 1 also tells us that this problem can be naturally overcome in our setting: to restore the *modularity* of the extensions and obtain full intuitionistic logic $\mathbf{J}$, we just need to consider the system $\mathbf{JR}$ (*intuitionistic* $\mathbf{R}$), which we obtain from $\mathbf{R}$ by substituting $\amalg Ec$ with $\amalg Ei$, and deleting $**c$. Indeed, $\mathbf{JR}$ is an intermediate system between $\mathbf{R}^+$ and $\mathbf{R}$, $\mathbf{R}^+ \subset \mathbf{JR} \subset \mathbf{R}$, and we can extend it with *int* to obtain full intuitionistic logic:

Proposition 7 *Adding the rule int to* $\mathbf{JR}$ *yields* $\mathbf{J}$.

We show this by (i) proving that R reduces to a partial order, and (ii) that relevant $\to$, $\neg$, $\amalg$ and the corresponding relevance rules reduce to intuitionistic $\supset$, $\sim$, $\bot$ and the corresponding intuitionistic rules.

To finish this section we give an example of derivation, i.e. we show that $G{:}\neg\neg A \to A$ is provable in $\mathbf{R}$ (note the use of $\amalg Ec$ and $**c$):

$$\dfrac{\dfrac{\dfrac{[c^*{:}\neg A]^1 \quad \dfrac{[b{:}\neg\neg A]^2 \quad [RGbc]^2}{c{:}\neg\neg A}\ monl}{}\ \neg E}{\dfrac{RGc^{**}c}{}\ **c \quad \dfrac{d{:}\amalg}{c^{**}{:}A}\ \amalg Ec^1}\ monl}{\dfrac{c{:}A}{G{:}\neg\neg A \to A}\ \to I^2}$$

This proof, formalized in Isabelle, is given in Section 5.

3 CORRECTNESS OF THE PRESENTATIONS

In this section we show that every non-classical logic $L = \mathcal{B} + \mathcal{T}$ is sound and complete with respect to the corresponding Kripke-style semantics.

Let us first extend the definition of $\models^{\mathsf{M}}$ as follows: $\models^{\mathsf{M}} \Delta$ means that $\models^{\mathsf{M}} R_i\, a\, a_1 \ldots a_n$ for all $R_i\, a\, a_1 \ldots a_n \in \Delta$; $\models^{\mathsf{M}} (\Gamma, \Delta)$ means that $\models^{\mathsf{M}} \varphi$ for all lwffs or rwffs $\varphi \in \Gamma \cup \Delta$; $\Delta \models R_i\, a\, a_1 \ldots a_n$ means that $\models^{\mathsf{M}} \Delta$ implies $\models^{\mathsf{M}} R_i\, a\, a_1 \ldots a_n$ for any model M; and $\Gamma, \Delta \models a{:}A$ means that $\models^{\mathsf{M}} (\Gamma, \Delta)$ implies $\models^{\mathsf{M}} a{:}A$ for any model M.

The explicit embedding of properties of the models, and the possibility of explicitly reasoning about them, via rwffs and relational rules, require us to consider also soundness and completeness for rwffs, where we show that $\Delta \vdash_L R_i\, a\, a_1 \ldots a_n$ iff $\Delta \models R_i\, a\, a_1 \ldots a_n$.

Definition 8 Let Γ be a set of lwffs and Δ a set of rwffs. The logic L is *sound* iff (i) $\Delta \vdash_L R_i\, a\, a_1 \ldots a_n$ implies $\Delta \models R_i\, a\, a_1 \ldots a_n$, and (ii) $\Gamma, \Delta \vdash_L a{:}A$ implies $\Gamma, \Delta \models a{:}A$. L is *complete* iff the converses hold.

Theorem 9 *$L = \mathcal{B} + \mathcal{T}$ is sound and complete.*

Some remarks. The proof is modular, since it is parameterized over the relational theory extending $\mathcal{B}$. Soundness follows by induction on the structure of the derivations. Completeness follows by a modification of standard techniques: we build a canonical model, $\mathsf{M}^C = (\mathsf{W}^C, \mathsf{G}^C, \mathsf{R}^{uC}, \mathsf{R}^{eC}, *^C, \mathsf{V}^C)$, which provides a countermodel to underivable formulae, i.e. $\Delta \nvdash_L R_i\, a\, a_1 \ldots a_n$ implies $\Delta \not\models^{\mathsf{M}^C} R_i\, a\, a_1 \ldots a_n$, and $\Gamma, \Delta \nvdash_L a{:}A$ implies $\Gamma, \Delta \not\models^{\mathsf{M}^C} a{:}A$. We exploit the additional information on labels and relations between them contained in Δ_L, the deductive closure of Δ with respect to the logic L, to (i) prove completeness for rwffs, and (ii) build, for $\Gamma, \Delta \nvdash_L a{:}A$, a 'global' set of labelled formulae which is *maximal* with respect to $a{:}A$,[11] instead of separately building the usual maximal sets of unlabelled formulae. W^C (including $\mathsf{G}^C, *^C$) is then obtained by partitioning the resulting maximal set of lwffs with respect to the labels, and $\mathsf{V}^C(b^C, B) = 1$ iff $b{:}B$ is contained in it. Finally, we do not adopt the standard definition of R^{uC}, but define $(a, a_1, \ldots, a_u) \in \mathsf{R}^{uC}$ iff $R^u\, a\, a_1 \ldots a_u \in \Delta_L$. Similarly for R^{eC}.

3.1 POSITIVE FRAGMENTS AND INTERRELATED RELATIONS

In Section 2 we argued that an unrestricted *monl* rule produces an unsound system in which intuitionistic and classical implication are equivalent, and that soundness is regained when applications of *monl* are restricted to persistent formulae. We show now that

[11] The pair (Γ, Δ_L) is maximal with respect to $a{:}A$ when '$b{:}B \notin \Gamma$ iff $\Gamma \cup \{b{:}B\}, \Delta_L \vdash_L a{:}A$'.

the correctness of our presentations (Theorem 9) depends on another restriction we have imposed in Section 2, that there are no a priori assumptions on the interrelationships of the different relations associated with universal and existential modalities. If this restriction is withdrawn and the relations are interrelated, then incompleteness may arise.

To illustrate this, we consider the positive fragments of (classical) modal logics. Without negation we cannot define $\Diamond$ in terms of $\Box$ and derive the rules for $\Diamond$. Indeed, there need be no a priori reason why $\Box$ and $\Diamond$ are related at all. Therefore, we characterize the positive fragments containing both $\Box$ and $\Diamond$ by the interrelationships between $R^{\Box}$ and $R^{\Diamond}$, which are specified by a (possibly empty) collection of the following Horn relational rules:

$$\frac{x\,R^{\Diamond}\,y}{x\,R^{\Box}\,y}\ (\Diamond\Box) \qquad \frac{x\,R^{\Box}\,y}{x\,R^{\Diamond}\,y}\ (\Box\Diamond)$$

Then, using these rules, we can prove theorems stating relationships between $\Box$ and $\Diamond$. For instance, using $(\Diamond\Box)$ we can prove

$$x{:}(\Diamond A \wedge \Box B) \supset \Diamond(A \wedge B)\,, \tag{23}$$

and using $(\Box\Diamond)$ we can prove

$$x{:}(\Diamond A \supset \Box B) \supset \Box(A \supset B)\,. \tag{24}$$

That these theorems are provable is not surprising: correspondence theory provides a means of showing that (23) corresponds to the semantic condition $\mathsf{R}^{\Diamond} \subseteq \mathsf{R}^{\Box}$, and that (24) corresponds to $\mathsf{R}^{\Box} \subseteq \mathsf{R}^{\Diamond}$.

Now consider

$$x{:}\Box(A \vee B) \supset (\Diamond A \vee \Box B)\,, \tag{25}$$

which corresponds, in the above sense, to $\mathsf{R}^{\Box} \subseteq \mathsf{R}^{\Diamond}$, and therefore is true in the models satisfying this property. By analysis of normal form proofs (cf. Section 4), we can show that (25) is not provable using $(\Box\Diamond)$.[12] This illustrates that:

Theorem 10 *If the R_is associated with the modalities are not independent, then there are positive fragments of non-classical logics that are incomplete with respect to the corresponding Kripke-style semantics.*

An analogous problem holds for Hilbert presentations, as pointed out by Dunn in [9]; he ensures the completeness of the 'absolutely' positive fragment of modal logic (i.e. without negation and implication) by extending his Hilbert-style deductive system with postulates equivalent to (23) and (25). Similarly, we could restore completeness in our setting by giving up our claim to a fixed base logic extended with relational theories, and adding a rule directly encoding (25), e.g.

$$\frac{}{x{:}\Box(A \vee B) \supset (\Diamond A \vee \Box B)}$$

However, such a rule is not in the spirit of ND since it does not contribute to the theory of meaning of the connectives. Moreover, it complicates proof normalization arguments.

4 NORMALIZATION

Following Prawitz [24, 25], we can show that our presentations have the following properties:

Theorem 11
(I) *The deductive machinery is minimal: the proof systems formalize the minimum fragment of first-order logic required by the semantics.*
(II) *Derivations are rigorously partitioned: the derivation of lwffs may depend, via rules for non-local connectives, on derivations of rwffs, but not vice versa.*
(III) *Derivations normalize: the derivations of lwffs have a well-structured normal form that satisfies the subformula property.*

For comparison, consider the semantic embedding approach (e.g. [19]), in which a non-classical logic is encoded as a 'suitable' (e.g. intuitionistic or classical) first-order theory by axiomatizing an appropriate definition of truth: (i) a non-classical logic constitutes a theory of full first-order logic, as opposed to an extension of labelled propositional logic with Horn-clauses; (ii) all structure is lost as propositions and relations are flattened into first-order formulae; (iii) there are normal forms, those of ND for first-order logic, but derivations of lwffs are mingled with derivations of rwffs, as opposed to the separation between the base logic and the relational theory that we have enforced.

This separation is in the philosophical spirit of LDSs, and it also provides extra structure that is pragmatically useful: since derivations of rwffs use only the resources of the relational theory, we may be able to employ theory-specific reasoners successfully to automate proof construction.[13] However, in exchange for this extra structure there are limits to the generality of the formulation: the properties in Theorem 11 depend on design decisions we have made, in particular, the

[12] This is because the proof of (25) requires properties of classical negation. Thus, instead of 'strengthening' the proof system, we could try to restore completeness by adopting a semantics with a 'weaker' negation.

[13] Then, to further restrict the structure of normal derivations, it is interesting to study the eliminability of *monl* from the systems.

use of Horn relational theories. This, of course, places stronger limitations on what we can formalize than a semantic embedding in first-order logic. Consider, for instance, the relevance logic **RM**, the extension of **R** with the postulate

$$Rabc \text{ implies } (RGac \text{ or } RGbc), \qquad (26)$$

which corresponds to the axiom $A \to (A \to A)$. We cannot formalize **RM** because (26) is not formalizable as a set of Horn rules. This is a design decision. Consider the alternatives. We can extend our deductive machinery by providing proof rules for a full first-order relational theory and explicitly add (26) as an axiom. However, if we then maintain (II) of Theorem 11 we lose completeness (with respect to the semantics), since, by analysis of normal form proofs, we can show that $G{:}A \to (A \to A)$ is not provable. Alternatively, we can regain completeness by giving up (II), by identifying *falsum* in the first-order relational theory with $\perp\!\!\!\perp$. However, the resulting system is then essentially equivalent to semantic embedding and we lose (I); cf. [2], where we investigated analogous problems for modal logics.

But there is also another reason why this latter solution is not satisfactory: since it is based on the $\perp\!\!\!\perp$ rules, it does not apply for positive fragments. For these (and also for full logics), we can regain completeness by again giving up (II) to introduce rules similar to Simpson's geometric rules [31], e.g. we encode (26) with the 'relational' rule:

$$\frac{Rabc \qquad \begin{array}{c}[RGac]\\ \vdots\\ d{:}A\end{array} \qquad \begin{array}{c}[RGbc]\\ \vdots\\ d{:}A\end{array}}{d{:}A}$$

5 IMPLEMENTATION AND ITS CORRECTNESS

We have implemented the work described in this paper in the Isabelle system [23], which is based on a logical framework of minimal implicational logic with quantification over higher types [22]. Since the implementation issues are not significantly different from the simpler case for modal logic described in [2], we refer the interested reader there (and also [3]) for more extensive details and we give only a brief overview here.

5.1 THE IMPLEMENTATION

We call the framework logic of Isabelle $\mathcal{M}$, and write universal quantification and implication in 'machine readable' form as `!!` and `==>`.

A logic is encoded in Isabelle using a theory composed of a signature and axioms, which are formulae in the language of $\mathcal{M}$. The axioms are used to establish the validity of judgements, which are assertions about syntactic objects declared in the signature [15]. Then proving theorems in the encoded logic simply means proving theorems with these axioms in the metalogic. As an example, consider the theory $\mathcal{M}_{\mathbf{R}}$,

```
R = Pure +    - R extends Pure (Isabelle's metalogic)
types         - with the following signature and axioms
  l,o 0
arities
  l, o :: logic
consts
  G     :: "l"
  inc   :: "o"
  ...
  (* Connectives *)
  imp   :: "[o, o] => o"          (infixr 25)
  star  :: "l => l"               ("_*" [40] 40)
  neg   :: "o => o"               ("~_" [40] 40)
  ...
  (* Judgements *)
  L :: "[l, o] => prop"      ("(_ : _)" [0,0] 100)
  R :: "[l, l, l] => prop"
                          ("(R _ _ _)" [0,0,0] 100)
rules
  (* Base Logic *)
  impI  "[| !!b c. [| b:A; R a b c |] ==> c:B |]
                                    ==> a:(A imp B)"
  monl  "[| a:A; R G a b |] ==> b:A"
  negE  "[| a: ~A; a* : A |] ==> b : inc"
  incEc "(a: ~A ==> b : inc) ==> a* : A"
  ...
  (* Properties of R *)
  starc "R G a** a"
  ...
end
```

which encodes the presentation of **R** given in Definition 6 (we elided some of the details). The signature of $\mathcal{M}_{\mathbf{R}}$ declares two types `l` and `o`, for labels and (unlabelled) formulae. Constants and connectives are then declared as typed constants over this signature; e.g. `inc` (for incoherence, i.e. $\perp\!\!\!\perp$) of type `o`, and `neg` of type `o => o`. There are two judgements, encoded as predicates: first, `L a A`, for provable lwffs, which we abbreviate to `a:A`; second, `R a b c`, for provable rwffs. The axioms for `L` and `R` correspond directly to the rules in Definition 6. Note that in the axioms, free variables are implicitly outermost universally quantified, and there is additional information present to fix notation and help Isabelle's parser.

Using this encoding we can prove `G : ~~A imp A` as follows (cf. the proof of $G{:}\neg\neg A \to A$ given in Section 2.4):

```
> goal R.thy "G : ~~A imp A";  - We enter the goal
G : ~~A imp A          - Isabelle responds with the goal
 1. G : ~~A imp A      - and the subgoal(s) to be proved

> by (rtac impI 1);      - Resolve the goal with impI
G : ~~A imp A
 1. !!b c. [| b : ~~A; R G b c |] ==> c : A

> by ((rtac monl 1) THEN  (rtac starc 2));
                        - Apply monl and dispose of
G : ~~A imp A           - second subgoal using starc
 1. !!b c. [| b : ~~A; R G b c |] ==> c** : A

> by (EVERY [rtac incEc 1, rtac negE 1, atac 2]);
G : ~~A imp A
 1. !!b c. [| b : ~~A; R G b c; c* : ~A |]
                                      ==> c : ~~A

> by (rtac monl 1);
G : ~~A imp A
 1. !!b c. [| b : ~~A; R G b c; c* : ~A |]
                                ==> ?a5(b, c) : ~~A
 2. !!b c. [| b : ~~A; R G b c; c* : ~A |]
                                ==> R G ?a5(b, c) c

> by (REPEAT (atac 1));  - Instantiate metavariables
G : ~~A imp A    - to solve the subgoals by assumption.
No subgoals!     - Isabelle tells us we are finished!
```

5.2 CORRECTNESS

By reasoning about our encoding and the metalogic $\mathcal{M}$ we can prove that $\mathcal{M}_{\mathbf{R}}$ corresponds to the original $\mathbf{R}$. We do this in two parts, by showing first *adequacy*, that any proof in $\mathbf{R}$ can be reconstructed in $\mathcal{M}_{\mathbf{R}}$, and then *faithfulness*, that we can recover from any derivation in $\mathcal{M}_{\mathbf{R}}$ a proof in $\mathbf{R}$ itself.

Adequacy is easy to show, because the rules of $\mathbf{R}$ map directly onto the axioms of $\mathcal{M}_{\mathbf{R}}$. A simple inductive argument on the structure of proofs in $\mathbf{R}$ establishes this (cf. [2, §5.2]). Faithfulness is more complex, since there is no such simple mapping in this direction: arbitrary derivations in $\mathcal{M}_{\mathbf{R}}$ do not map directly onto proofs in $\mathbf{R}$. Instead we use proof-theoretic properties of $\mathcal{M}$: any derivation in $\mathcal{M}$ is equivalent to another in a normal form. Hence, given a derivation in $\mathcal{M}_{\mathbf{R}}$ we can, by induction over its normal form, find a derivation in $\mathbf{R}$. This establishes faithfulness (again cf. [2, §5.2]). Moreover, this proof is constructive: it not only tells us that there is a proof in $\mathbf{R}$, it provides an effective method for finding one.

6 RELATED AND FUTURE WORK

Gabbay has proposed LDSs as a general methodology for presenting logics [12]. The formal details are different from our proposal. For example, labelled modal logics presented in [12] are based on a notion of diagrams and logic data-bases, which are manipulated by complex multiple conclusion rules. The result is similar to semantic embedding (cf. [2]), to which we have compared our work above. In [4] labelled tableaux for substructural logics are proposed based on algebraic semantics. The rules support automated proof search, but are not easy to recast as ordinary pure ND proof rules (e.g. their general closure rule depends on arbitrarily many formulae).

In [20, 21], Orlowska introduces tableaux-like relational proof systems for relevance, modal and intuitionistic logics, by first translating formulae into suitable relations, and then proving a formula by decomposing its relational translation into simpler relations. Although the metalogic is different, relational logic instead of predicate logic, this method is comparable to semantic embedding, since formulae of the logic and relations from the Kripke semantics are treated in a uniform way as relations.

Our work is closely related to, and influenced by, the algebraic approach proposed by Dunn (cf. [8] and the references there). Dunn introduces *gaggle theory* as an abstraction of Boolean algebras with operators [18], where n-ary operators are interpreted by means of $n+1$-ary relations. Gaggle theory yields a landscape of algebras where the standard Kripke semantics for a particular logic is obtained by manipulating the gaggle presentation at the level of the canonical model, as opposed to instantiating the appropriate relational theory as in our approach. For instance, an analysis of the canonical model shows how to reduce the ternary relation associated with the binary intuitionistic implication to the more customary partial order on possible worlds. This algebraic approach is extremely powerful, but does not lend itself well to direct implementation; however, with appropriate simplifications or by combination with Belnap's *display logic* (as in [26]) this may be possible. We plan to investigate this as future work.

Acknowledgements

We thank Andreas Nonnengart for helpful discussions.

References

1. A. Avron. Simple consequence relations. *Information and Computation*, 92:105–139, 1991.

2. D. Basin, S. Matthews, and L. Viganò. Labelled propositional modal logics: Theory and practice. Technical Report MPI-I-96-2-002, Max-Planck-Institut für Informatik, Saarbrücken, 1996. Available at the URL http://www.mpi-sb.mpg.de/~luca/Publications/publications.html.

3. D. Basin, S. Matthews, and L. Viganò. Natural deduction for non-classical logics. Technical Report MPI-I-96-2-006, Max-Planck-Institut für Informatik, Saarbrücken, 1996. Available at the URL http://www.mpi-sb.mpg.de/~luca/Publications/publications.html.

4. M. D'Agostino and D. Gabbay. A generalization of analytic deduction via labelled deductive systems. part I : Basic substructural logics. *Journal of Automated Reasoning*, 13:243–281, 1994.

5. K. Dosen. Negation as a modal operator. *Reports on Mathematical Logic*, 20:15–27, 1986.

6. J. M. Dunn. Relevance logic and entailment. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume III. Reidel, Dordrecht, 1986.

7. J. M. Dunn. Star and perp: Two treatments of negation. In J. Tomberlin, editor, *Philosophical Perspectives*, volume 7. Ridgeview, Atascadero, 1994.

8. J. M. Dunn. Gaggle theory applied to modal, intuitionistic, and relevance logics. In I. Max and W. Stelzner, editors, *Frege Kolloquim 93*. de Gruyter, Berlin, 1995.

9. J. M. Dunn. Positive modal logic. *Studia Logica*, 55:301–317, 1995.

10. L. Fariñas Del Cerro and A. Herzig. Combining classical and intuitionistic logic. In *Proceedings of FroCoS'96*. Kluwer, 1996.

11. M. Fitting. *Proof Methods for Modal and Intuitionistic Logics*. Kluwer, Dordrecht, 1983.

12. D. Gabbay. LDS - labelled deductive systems, volume 1 - foundations. Technical report, Max-Planck-Institut für Informatik, Saarbrücken, 1994.

13. P. Gardner. Equivalences between logics and their representing type theories. *Mathematical Structures in Computer Science*, 5:323–349, 1995.

14. R. Goldblatt. Semantical analysis of orthologic. *Journal of Philosophical Logic*, 3:19–35, 1974.

15. R. Harper, F. Honsell, and G. Plotkin. A framework for defining logics. *Journal of the ACM*, 40(1):143–184, 1993.

16. G. Hughes and M. Cresswell. *A companion to modal logic*. Methuen, London, 1984.

17. I. Humberstone. Interval semantics for tense logic: Some remarks. *Journal of Philosophical Logic*, 8:171–196, 1979.

18. B. Jónsson and A. Tarski. Boolean algebras with operators. *American Journal of Mathematics*, 73–4:891–939, 127–162, 1951-52.

19. H.-J. Ohlbach. Translation methods for non-classical logics: an overview. In *Bulletin of the IGPL*, volume 1, Saarbrücken, 1993.

20. E. Orlowska. Relational interpretation of modal logics. In H. Andréka, J. D. Monk, and I. Németi, editors, *Algebraic Logic*. North-Holland, Amsterdam, 1991.

21. E. Orlowska. Relational proof system for relevant logics. *Journal of Symbolic Logic*, 57(4):1425–1440, 1992.

22. L. Paulson. The foundation of a generic theorem prover. *Journal of Automated Reasoning*, 5:363–397, 1989.

23. L. Paulson. *Isabelle: A Generic Theorem Prover*. Springer, Berlin, 1994.

24. D. Prawitz. *Natural Deduction, a Proof-Theoretical Study*. Almqvist and Wiksell, Stockholm, 1965.

25. D. Prawitz. Ideas and results in proof theory. In J. E. Fensted, editor, *Proceedings of the 2nd Scandinavian Logic Symposium*. North-Holland, 1971.

26. G. Restall. Display logic and gaggle theory. Technical Report TR-ARP-22-95, Australian National University, Dec. 7, 1995.

27. G. Restall. Negation in relevant logics (how I stopped worrying and learned to love the Routley star). Technical Report TR-ARP-3-95, Australian National University, Feb. 20, 1995.

28. R. Routley and R. Meyer. The semantics of entailment – II. *Journal of Philosophical Logic*, 1:53–73, 1972.

29. R. Routley, V. Plumwood, R. Meyer, and R. Brady. *Relevant Logics and their Rivals*. Ridgeview, Atascadero, California, 1982.

30. P. Schroeder-Heister. A natural extension of natural deduction. *Journal of Symbolic Logic*, 49(4):1284–1300, 1984.

31. A. Simpson. *The Proof Theory and Semantics of Intuitionistic Modal Logic*. PhD thesis, University of Edinburgh, Edinburgh, 1993.

32. G. Sundholm. Systems of deduction. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume I. Reidel, Dordrecht, 1983.

33. G. Sundholm. Proof theory and meaning. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume III. Reidel, Dordrecht, 1986.

34. J. van Benthem. Correspondence theory. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume II. Reidel, Dordrecht, 1984.

35. J. van Benthem. *Modal Logic and Classical Logic*. Bibliopolis, Napoli, 1985.

"Statistical" First Order Conditionals

Ronen I. Brafman
Computer Science Department
University of British Columbia
Vancouver, B.C., V6T 1Z4, Canada
brafman@cs.ubc.ca
http://www.cs.ubc.ca/spider/brafman/home.html

Abstract

A first-order conditional logic is defined in which conditionals such as $\alpha \rightarrow \beta$ are interpreted as saying that most/normal/typical objects which satisfy α satisfy β as well. This qualitative statistical interpretation is achieved by imposing additional structure on the domain of a *single* first-order model in the form of an ordering over domain elements and tuples. $\alpha \rightarrow \beta$ then holds if all objects with property α whose ranking is minimal satisfy β as well. These minimally ranked objects represent the typical or common objects having the property α. This semantics differs from that of the more common subjective interpretation of conditionals over *sets* of standard first-order structures, and it provides a more natural way of modeling qualitative statistical statements, such as "most birds fly," or "normal birds fly." We provide a sound and complete axiomatization of this logic as well as a probabilistic semantics for it.

1 INTRODUCTION

Conditional logics have been the focus of much AI research in recent years because of their important connection to default reasoning and belief revision. Most work on conditional logic has concentrated on the propositional case, for example (Lewis, 1973; Boutilier, 1994; Kraus, Lehmann, & Magidor, 1990; Lehmann & Magidor, 1992). First-order conditional logics have received less attention, and there does not seem to be agreement on their formulation. This is understandable given that first-order logic is considerably more expressive than propositional logic. This additional expressivity requires one to deal with issues that do not arise in the propositional case, such as the interaction between quantifiers and the conditional operator.

The language of propositional conditional logics contains, in addition to the standard boolean operators, a binary conditional operator $\rightarrow$. These logics are interpreted over a structure that consists of an ordered *set* S of standard propositional models. The formula $\alpha \rightarrow \beta$ is satisfied by such a structure when all those standard propositional models in S that are minimal elements of the set of standard models satisfying α, satisfy β as well. While this is just a rough description, for instance, the ordering may be relative to each world or partial, it captures the essence of the semantics. This semantics justifies an intuitive reading of $\alpha \rightarrow \beta$ as "in the most normal α worlds, β holds."

This semantics can generalized to the first order case using a similar interpretation over an ordered set of standard first-order models. There are various issues that this interpretation raises. For instance, should the domain of all these models be the same? Or, should constants be interpreted as rigid designators? Regardless of these choices, the essence of the interpretation is the same, and as before, an informal reading of the conditional $\alpha \rightarrow \beta$ would be "in the most normal α worlds, β holds." The main difference is that now, α and β are part of a much richer language.

However, the structure of first-order models provides an alternative interpretation for conditionals. First-order models describe the properties of many individuals comprising the domain of the model. This is to be contrasted with propositional models which can be viewed as describing the properties of a single object. Hence, in a sense, first-order structures define a set of propositional structures, each corresponding to the properties a single object in the domain of the first-order structure. Thus, another possible interpretation of first-order conditional logics is w.r.t. a *single* first-order structure that is supplemented with an ordering over *domain elements*. Now, $\alpha \rightarrow \beta$ is interpreted as saying that "all objects in the structure's domain that are normal w.r.t. the property α satisfy β as well."

There are different manners in which one can enhance a first-order structure. In particular, our choice of an ordering relation over the structure's domain is not the only one: Schlechta's work on defaults as generalized quantifiers (Schlechta, 1995) adds weak filters to a first-order structure to obtain an interpretation of defaults (or conditionals) that is similar in flavor to ours. Friedman, Halpern, and Koller (1996) use plausibility measures (Friedman & Halpern, 1995) to assign degrees of plausibility to different domain elements. We discuss the view of first-order conditionals as generalized quantifiers in a longer version of this paper, and we discuss Schlechta's and Friedman, Halpern and Koller's work in Section 5.

The two approaches to first-order conditional logics outlined above closely resemble two interpretations of probabilistic statements: the frequentist interpretation and the subjectivist interpretations. While these interpretations have a long history, two recent works on probabilistic logic brought them to the attention of the formal reasoning community in AI. Bacchus (1990) and Halpern (1990) examine two types of statements about probabilities: statements about subjective beliefs, such as "the probability that Tweety flies is 0.9," and statistical statements, such as "90% of birds fly." They suggest that the first type of statement is naturally modeled by a probability measure over a possible-worlds structure, while the second statement can be interpreted by imposing a probability assignments over a single domain. The choice between these interpretation is akin to the choice between the two possible interpretations of first-order conditional. Indeed, conditional sentences can be viewed as qualitative counterparts of probabilistic sentences. This is a consequence of the probabilistic semantics of conditionals provided by Adams (1975) and by Goldszmidt and Pearl (1992). According to this semantics, and depending on which interpretation of probabilities one chooses, $\alpha \rightarrow \beta$ can be roughly understood as a statement of subjective belief, i.e., "in those α worlds I consider most likely, β is the case," or as qualitative statistical statements, i.e., "in the actual world, most objects with property α have property β as well."

In this paper, we formalize this second, statistical interpretation of first-order conditional logics. We interpret conditionals over a standard first-order structure to which we add an ordering over domain elements. More precisely, for every natural number n, there is a total pre-order over the set of n-tuples of domain elements. A conditional of the form $\varphi \rightarrow_{(\bar{x})} \psi$ is interpreted as saying that all the minimal tuples $\bar{c}$ of length $|\bar{x}|$ that satisfy φ when $\bar{c}$ is substituted for $\bar{x}$, must also satisfy ψ under this substitution. A natural extension of this semantics imposes restrictions on the possible pre-orders and on the relationship between pre-orders of tuples of different lengths. We explore three variants of this semantics and show that the most powerful of them can be given a probabilistic semantics akin to Goldszmidt and Pearl's (1992) ϵ-semantics.

Our work is closely related to Bacchus's work on representing statistical information in first-order logic (Bacchus, 1990). Our logic can be thought of as a qualitative counterpart to Bacchus's quantitative approach. Indeed, the form of quantification we use resembles Bacchus' notation, and some of our axioms can be viewed as qualitative counterparts of similar axioms that appear in his work. However, the difference between the qualitative and quantitative statements of our logics leads to an important technical and expressive difference. Bacchus's logic has numbers as objects in the language, and he allows quantification over them as well as various operations, such as addition and multiplication. Our counterpart of probabilities, ranks, are not objects in the language, and we do not manipulate them within the logic.

Once the distinction between subjective and statistical conditionals is understood, it can help us approach problems such as the lottery paradox. It is perfectly consistent to state that most (or normal) tickets will not win the lottery, yet some ticket will win. This has been called a paradox because in some subjective conditional logics, e.g., Delgrande (1987), the following theory is inconsistent:

$$\forall x(true \rightarrow \neg Winner(x)) \ , \ true \rightarrow \exists x Winner(x).$$

Interpreted subjectively, the first sentence says that in our most normal worlds, all tickets will not win, while the second sentence claims that in our most normal worlds a winner exists. This, of course, is a contradiction. Moreover, the first statement does not seem to capture our information about lotteries. The statistical statement that most tickets will not win the lottery is a more plausible one. More importantly, we see that the two interpretations are fundamentally different.

Another example of a theory that is inconsistent under a subjective interpretation is the following:

$$\forall x, y(Pet(x, y) \rightarrow (Dog(y) \vee Cat(y)),$$

$$\forall y(Pet(John, y) \rightarrow (Snake(y)).$$

The subjective interpretation of this theory is that in our most normal worlds, everybody's pets are either dogs or cats, while John's pets are snakes. But clearly, John is someone, so that, based on the first sentence,

his pets should be dogs or cats. A statistical interpretation makes more sense here. Under such an interpretation, we would understand these formulas as saying that most people's pets are dogs or cats, but most of John's pets are snakes. Both statements are consistent under the qualitative statistical semantics we present.

In the Section 2 we provide a formal account of the semantics of first-order statistical conditional logic. In Section 3, we describe a sound and complete axiomatization of this logic. In Section 4, we show that our conditional logic can be given probabilistic semantics. We end with a discussion of the logic and its relation with other work, in Section 5. The proof of our main technical result appears in Appendix A. A longer version of this paper[1] contains all the remaining proofs, additional discussion of related work, and shows that the conditional operator can be replaced by a class of quantifiers or alternatively, by a set of new predicates. This latter formulation leads to a semi-standard first-order language with a much simpler axiom system. Throughout this paper we assume familiarity with the basic syntax and semantics of first-order logic.

2 LANGUAGE AND SEMANTICS

We now proceed with a description of the language of statistical conditional logic, a language which extends the language of first-order logic (FOL). This will be followed by a definition of the language's formal semantics.

We assume the existence of an underlying first-order language $\mathcal{L}$, containing (possibly infinite) sets of predicate symbols P, constants symbols C, function symbols F, and variables V. Our language, $\mathcal{L}_C$ consists of a set of well formed formulas (wff) defined below using P, C, F and V.

Definition 1 *The set of* well formed formulas *of $\mathcal{L}_C$ is defined inductively as follows:*

- *Atomic formulas of $\mathcal{L}_C$ are wffs (an atomic formula is a predicate symbol with an appropriate number of terms).*
- *If φ and ψ are wffs, then so are $\neg\varphi, \varphi \Rightarrow \psi$.*
- *If φ is a wff and x is a variable, then $\forall x\varphi$ is a wff.*
- *If φ and ψ are wffs then $\varphi \rightarrow_{(\bar{x})} \psi$ is a wff.*

We shall use the symbols $\vee, \wedge, \Leftrightarrow$ and $\exists$ freely, with the understanding that they are defined by $\neg, \Rightarrow$ and $\forall$. We use the shorthand $\bar{x}$ for a sequence of variables $x_1, \ldots, x_n$ for some fixed n, when n is clear from the context.

Intuitively, the conditional $\varphi \rightarrow_{(\bar{x})} \psi$ can be read as saying that most, or that normal tuples that can be substituted for $\bar{x}$ and have the property φ, have the property ψ as well. Hence, in this formula, the variables $\bar{x}$ are implicitly quantified by the conditional operator.

Let us consider a number of examples of wffs and their intuitive interpretation.

1. $\mathit{friend}(x,y) \rightarrow_{(x,y)} \mathit{like}(x,y)$. Here, x and y are implicitly quantified by the conditional operator. This sentence can be read as "for most pairs (x,y) such that y is x's friend, x likes y." Another possible reading would be "all normal (x,y) pairs such that y is x's friend are such that x likes y."
2. $\forall x(\mathit{friend}(x,y) \rightarrow_{(y)} \mathit{like}(x,y))$. Here x is universally quantified, while y is implicitly quantified by the conditional operator. This sentence can be read as "everyone likes most of their friends."
3. $\forall y(\mathit{friend}(x,y) \rightarrow_{(x)} \mathit{like}(x,y))$ can be read as "all people are liked by most of their friends."
4. $(\mathit{friend}(x,y) \rightarrow_{(y)} \mathit{like}(x,y)) \rightarrow_{(x)} \mathit{tall}(x)$ can be read as "most people who like most of their friends are tall."

Next, we extend the standard definition of free-variables to $\mathcal{L}_C$.

Definition 2 *We define the variables of a term as follows:*

- *If $t = v \in V$ then $var(t) = v$.*
- *If $t = f(t')$ then $var(t) = var(t')$.*
- *If $t = c \in C$ then $var(t) = \emptyset$*

The free variables of a wff ψ ($fv(\psi)$) are defined inductively:

- *If $P(t_1, \ldots, t_n)$ is an atomic formula, then $fv(P(t_1, \ldots, t_n)) = \cup_{i=1}^{n} var(t_i)$.*
- *If φ and ψ are wffs, then $fv(\neg\varphi)$=$fv(\varphi)$ and $fv(\varphi \Rightarrow \psi)$=$fv(\varphi) \cup fv(\psi)$.*
- *If φ is a wff and x is a variable, then $fv(\forall x\varphi) = fv(\varphi) \setminus \{x\}$.*

[1] Available online at http://www.cs.ubc.ca/spider/brafman/publications.html.

- *If φ and ψ are wffs and x is a variable, then* $fv(\varphi \rightarrow_{(x)} \psi) = [fv(\varphi) \cup fv(\psi)] \setminus \{\bar{x}\}$

In order to define our models, we need the following:

Definition 3 $\mathbf{R}$ *is a* ranking function *on* D *if* $\mathbf{R} : D \mapsto \Omega$, *where* Ω *is a totally ordered set.*

We interpret the language $\mathcal{L}_C$ over a class of structures consisting of first-order models whose domain elements are ranked.

Definition 4 *A ranked first-order structure is a pair* $\mathcal{M}=\langle \mathbf{M}, \mathbf{R} \rangle$

- $\mathbf{M}$ *is a standard first order structure.*
- $\mathbf{R} = \{R_n | n \in N\}$, *where for each* n, R_n *is a ranking function on* $|\mathbf{M}|^n$.[2]

We shall use the shorthand notation $(a,b)=_{\mathbf{R}}(c,d)$ for $R_2((a,b)) = R_2((c,d))$ (and similarly for other relations). We shall also talk about minimal elements in $|\mathbf{M}|^n$, with the understanding that minimality is w.r.t. the ranking R_n.

Definition 5 *Let* $s : Vars \mapsto ||\mathcal{M}||$ *be an assignment function. We define the notion of* satisfiability *of a wff* α *under* s *in a ranked first-order structure* $\mathcal{M}=\langle \mathbf{M}, \mathbf{R} \rangle$, *written* $\mathcal{M} \models \alpha[s]$, *as follows:*

- *If* α *is atomic then* $\mathcal{M} \models \alpha[s]$ *if* $\mathbf{M} \models \alpha[s]$.
- *If* $\alpha = \neg\beta$ *then* $\mathcal{M} \models \alpha[s]$ *if* $\mathcal{M} \not\models \beta[s]$.
- *If* $\alpha = \beta \Rightarrow \gamma$ *then* $\mathcal{M} \models \alpha[s]$ *if either* $\mathcal{M} \not\models \beta[s]$ *or* $\mathcal{M} \models \gamma[s]$ *or both.*
- *If* $\alpha = \forall x \beta$ *then* $\mathcal{M} \models \alpha$ *if* $\mathcal{M} \models \alpha[s][^x_a]$ *for all* $a \in ||\mathcal{M}||$.
  ($[s][^x_d]$ is the same as s, except that it assigns d to x.)
- *If* $\alpha = (\beta \rightarrow_{(x)} \gamma)$ *then* $\mathcal{M} \models \alpha[s]$ *if for each* $\bar{d} \in min_x(\beta)$ *we have that* $\mathcal{M} \models \gamma[s][^{\bar{x}}_{\bar{d}}]$, *where* $min_x\beta$ *is the set of minimal elements of the set* $\{\bar{d} \in |\mathbf{M}|^{|\bar{x}|} : \mathcal{M} \models \beta[s][^{\bar{x}}_{\bar{d}}]\}$.

That is, let B be the set of all tuples such that when substituted for $\bar{x}$ in β, we have that β is satisfied. Then, $\beta \rightarrow_{(\bar{x})} \gamma$ is satisfied if all those tuples that are minimal members of B (w.r.t. $R_{|\bar{x}|}$) are such that γ would also be satisfied when they are substituted for $\bar{x}$. For example, $bird(x) \rightarrow_{(x)} fly(x)$ would be satisfied if all those domain objects in the extension of *bird* (often denoted by $bird^{\mathbf{M}}$) that are minimal w.r.t. R_1, are also in $fly^{\mathbf{M}}$.

It is possible for the set $min_x\beta$ to be empty. However, we would like to exclude this case. Following (Kraus et al., 1990), we consider only *smooth* rankings.

Definition 6 $\mathbf{R}$ **is smooth** *(w.r.t. $\mathcal{M}$) if for all β and $\bar{x}$, $\{\bar{d} \in |\mathbf{M}|^{|\bar{x}|} : \mathcal{M} \models \beta[^{\bar{x}}_{\bar{d}}]\}$ is empty, has a minimal element, or equals $|\mathbf{M}|^{|\bar{x}|}$.*[3] *We define* **NS** *to be the class of smooth ranked structures.*

We follow the convention that conditionals of the form $false \rightarrow_{(\bar{x})} \beta$ are always satisfied. Also, notice that a conditional such as $bird(Tweety) \rightarrow_{(x)} fly(Tweety)$ is satisfied iff the corresponding material implication, $bird(Tweety) \Rightarrow fly(Tweety)$, is satisfied. This is because the satisfiability of $bird(Tweety)$ and $fly(Tweety)$ does not depend on s. Hence, $bird(Tweety)$ is satisfied under one assignment function iff it is satisfied under all assignment functions.

It seems that an appropriate intuitive reading of $\alpha \rightarrow_{(\bar{x})} \beta$, given the above definition, would be "tuples normal w.r.t. the property α have the property β." As we show later, a reading of "most tuples satisfying α satisfy β as well" can be justified, but with "most" understood more as "virtually all."

So far, we have not imposed any additional requirements on the functions R_n. However, there are two properties which we consider quite natural for some interpretations of the ranking $\mathbf{R}$. The first requirement, *permutation* says that the ranking is indifferent to the order of the elements in the tuple. In essence, this implies that the ranking is defined over bags (or multisets) of domain elements, rather then over tuples. The second requirement, *concatenation*, says that preferences are closed under concatenation. That is, if we prefer $\bar{c}$ over $\bar{d}$ and $\bar{c}'$ over $\bar{d}'$, then we prefer $\bar{c} \circ \bar{c}'$ over $\bar{d} \circ \bar{d}'$.

Definition 7 *A ranked structure* $\mathcal{M}=\langle \mathbf{M}, \mathbf{R} \rangle$ *satisfies* permutation *if for every* $n \in N$ *and for every permutation* π *over* $\{1, \ldots, n\}$, *we have that* $R_n(d_1, \ldots, d_n) = R_n(d_{\pi(1)}, \ldots, d_{\pi(n)})$.
It satisfies concatenation *if* $R_n(\bar{c}) \geq R_n(\bar{d})$ *and* $R_n(\bar{c}') \geq R_n(\bar{d}')$, *then* $R_n(\bar{c} \circ \bar{c}') \geq R_n(\bar{d} \circ \bar{d}')$, *and if in addition* $R_n(\bar{c}) > R_n(\bar{d})$ *then* $R_n(\bar{c} \circ \bar{c}') > R_n(\bar{d} \circ \bar{d}')$.

[2] Recall that $|\mathbf{M}|$ is the domain of $\mathbf{M}$.

[3] This latter condition has no analogue in KLM's definition.

(1)	All instances of FOL tautologies	
(2)	$\alpha \rightarrow_{(\bar{x})} \alpha$	**(Reflexivity)**
(3)	$\forall \bar{y}(\alpha(\bar{y}) \Leftrightarrow \beta(\bar{y})) \Rightarrow ((\alpha \rightarrow_{(\bar{x})} \gamma) \Leftrightarrow (\beta \rightarrow_{(\bar{x})} \gamma))$	**(Left Logical Equiv.)**
(4)	$\forall \bar{y}(\alpha(\bar{y}) \Rightarrow \beta(\bar{y})) \Rightarrow (\gamma \rightarrow_{(\bar{x})} \alpha) \Rightarrow (\gamma \rightarrow_{(\bar{x})} \beta)$	**(Right Weakening)**
(5)	$(\alpha \rightarrow_{(\bar{x})} \beta) \wedge (\alpha \rightarrow_{(\bar{x})} \gamma) \Rightarrow (\alpha \wedge \beta \rightarrow_{(\bar{x})} \gamma)$	**(Cautious Monotony)**
(6)	$(\alpha \rightarrow_{(\bar{x})} \beta) \wedge (\alpha \rightarrow_{(\bar{x})} \gamma) \Rightarrow (\alpha \rightarrow_{(\bar{x})} (\beta \wedge \gamma))$	**(And)**
(7)	$(\alpha \rightarrow_{(\bar{x})} \gamma) \wedge (\beta \rightarrow_{(\bar{x})} \gamma) \Rightarrow ((\alpha \vee \beta) \rightarrow_{(\bar{x})} \gamma)$	**(Or)**
(8)	$(\alpha \rightarrow_{(\bar{x})} \gamma) \wedge \neg(\alpha \wedge \beta \rightarrow_{(\bar{x})} \gamma) \Rightarrow \alpha \rightarrow_{(\bar{x})} \neg\beta$	**(Rational Monotony)**
(9)	$\forall \bar{x}\beta \Rightarrow \alpha \rightarrow_{(\bar{x})} \beta$	**(Universal Weakening)**
(10)	$(\alpha \rightarrow_{(\bar{x})} \beta) \Rightarrow (\exists \bar{x}\alpha \Rightarrow \exists \bar{x}(\alpha \wedge \beta))$	**(Instantiation)**
(11)	$(\alpha \rightarrow_{(\bar{x})} \beta) \Rightarrow (\alpha \rightarrow_{(\bar{y})} \beta)[\frac{\bar{x}}{\bar{y}}]$ where $\bar{y}$ does not occur in α and β	**(Renaming)**
(12)	$\forall \bar{y}(\alpha \rightarrow_{(\bar{x})} \beta) \Leftrightarrow (\alpha \rightarrow_{(\bar{x})} \forall \bar{y}\beta)$ whenever $\bar{y} \cap (\mathrm{fv}(\alpha) \cup \bar{x}) = \emptyset$	**(Universal Interchange)**
(13)	$(\alpha \rightarrow_{(\bar{x},\bar{y})} \beta) \Leftrightarrow (\alpha \rightarrow_{(\bar{y},\bar{x})} \beta)$	**(Permutation)**
(14)	$(\alpha \leq_{\bar{x}} \beta) \wedge (\alpha' \leq_{\bar{y}} \beta') \Rightarrow ((\alpha \wedge \alpha') \leq_{\bar{x},\bar{y}} (\beta \wedge \beta'))$ where $\bar{y} \cap \bar{x} = \bar{x} \cap \mathrm{fv}(\alpha' \vee \beta') = \bar{y} \cap \mathrm{fv}(\alpha \vee \beta) = \emptyset$.	**(Weak-Concat)**
(15)	$(\alpha <_{\bar{x}} \beta) \wedge (\alpha' \leq_{\bar{y}} \beta') \Rightarrow ((\alpha \wedge \alpha') <_{\bar{x},\bar{y}} (\beta \wedge \beta'))$ where $\bar{y} \cap \bar{x} = \bar{x} \cap \mathrm{fv}(\alpha' \vee \beta') = \bar{y} \cap \mathrm{fv}(\alpha \vee \beta) = \emptyset$.	**(Strong-Concat)**
(16)	$(\alpha \wedge \alpha' \rightarrow_{(\bar{x},\bar{y})} \gamma) \Rightarrow (\alpha \rightarrow_{(\bar{x})} (\alpha' \rightarrow_{(\bar{y})} \gamma))$ where $\bar{y} \cap \bar{x} = \emptyset$ $\bar{x} \notin \mathrm{fv}(\alpha')$, and $\bar{y} \notin \mathrm{fv}(\alpha)$	**(Distribution)**
(17)	$\alpha \rightarrow_{(\bar{x},\bar{y})} ((\alpha \rightarrow_{(\bar{x})} \beta) \Rightarrow \beta)$ where $\bar{y} \cap \bar{x} = \emptyset$	**(Projection)**

Figure 1: Axiom Schema

3 A SOUND AND COMPLETE AXIOMATIZATION

This section describes a sound and complete set of axioms for the class **NS**. We use the following shorthand notations:

$$\alpha \leq_{\bar{x}} \beta \stackrel{\text{def}}{=} \neg((\alpha \vee \beta) \rightarrow_{(\bar{x})} \neg\alpha)$$

$$\alpha <_{\bar{x}} \beta \stackrel{\text{def}}{=} (\alpha \vee \beta \rightarrow_{(\bar{x})} \neg\beta)$$

Notice that $\alpha \leq_{\bar{x}} \beta$ is read as "some normal tuples w.r.t. the property $\alpha \vee \beta$ satisfy α," and $\alpha <_{\bar{x}} \beta$ is read as "all normal tuples w.r.t. the property $\alpha \vee \beta$ satisfy $\neg\beta$". As we shall see, the former implies that those tuples that are normal for β are no more normal than those tuples normal for α, while the latter implies that those tuples normal for α are strictly more normal than those tuples normal for β. For this reason, we use the $\leq$ and $<$ symbols.

The axiom schema of our logic are displayed in Figure 1. **Modus Ponens**

$$\frac{\alpha \ , \ \alpha \Rightarrow \beta}{\beta}$$

is the only inference rule.

Here are intuitive interpretations of some of these axiom schema.

Reflexivity: Normal α objects satisfy α, e.g., normal birds are birds.

LLE: normal objects for logically equivalent formulas have identical properties. Objects can now be thought of as normal w.r.t. a set.

RW & AND: properties that are common to all normal objects for some formula are closed under logical implication.

Cau. Monotony: if all normal α objects satisfy β than normal α objects are normal for $\alpha \wedge \beta$. For example, if Tweety is a normal bird and normal birds fly, then Tweety is a normal flying bird.

OR: normal objects for $\alpha \vee \beta$ are the union of normal objects for α and β.

Rat. Monotony: if normal α objects are γ , but some normal $\alpha \wedge \beta$ objects are not γ , then normal objects for α are not β . For example, if normal birds fly, but it is not the case that normal birds weighing over 50 Kg. fly, then normal birds weigh less than 50 Kg.

Weakening: if all objects have property β then, in particular, normal α objects satisfy β .

Instantiation: if there are α objects then there are normal α objects.

Renaming: invariance under renaming of the bound

variables.

Interchange: if we have a pair of normal and universal quantifiers quantifying over disjoint variables, then their order can be exchanged. For example, "for any color, normal children can name that color" if and only if "normal children can name all colors." This shows that there are certain problems with the interpretation of the conditional $\alpha \rightarrow_{(\bar{x})} \beta$ as "most α's are β." Indeed, it does not seem to be necessarily the case that if for any color, most children can name that color, then most children can name all colors. As we show later on, the "most" interpretation can be justified via a probabilistic interpretation of conditionals. However, this latter notion of "most" may be better represented by the expression "virtually all".

The role of Axiom (12) in the proof of the completeness theorem is worth mentioning. This axiom guarantees that the models we construct are smooth. Suppose that there is no minimal rank for tuples satisfying α and that the domain is infinite. We can have a model that satisfies $\forall y(\alpha \rightarrow_{(x)} \beta)$, where for each domain element d there is some rank r_d such that (1) there are elements e whose rank is lower than r_d such that $\alpha[^x_e]$ is satisfied, and (2) $(\alpha \Rightarrow \beta)[^{x,y}_{c,d}]$ is satisfied iff c belongs to a rank lower than r_d. If we construct the model so that$\{r_d : d \in \|\mathcal{M}\|\}$ forms an infinite descending chain, the wff $(\alpha \rightarrow_{(x)} \forall y\beta)$ will not be satisfied.

Permutation: one can permute the variables bounded by the conditional operator.

Weak-Concat: if tuples normal for α (resp. α') are as normal as tuples normal for β (resp. β') then tuples normal for $\alpha \wedge \alpha'$ are as normal as tuples normal for $\beta \wedge \beta'$. Thus, if Alice is a normal female and Bill is a normal male then (Alice,Bill) is a normal female-male pair.

Strong-Concat: is similar to (14), only with strict inequality.

Distribution: allows us to minimize components separately. Suppose that members of all normal female/male pair look differently. Hence, it is true that given a normal female, all normal males look different than her.

Projection: provides another form of minimization by component. Suppose that (c, d) is a normal pair w.r.t. the property α. If all objects that together with d are normal w.r.t. α satisfy β, than c should satisfy β as well. That is, fixing d as one component of the pair, c should be normal for α together with d if (c, d) is normal for α.

We can prove the following results:

Theorem 1 *Axioms (1)-(12) are sound w.r.t. the class* **NS**.

Theorem 2 *Axioms (13) is sound w.r.t the class of* **NS** *structures satisfying* permutation *and axioms (14)-(17) are sound w.r.t the class of* **NS** *structures satisfying* concatenation.

Theorem 3 *For a countable language $\mathcal{L}_C$, axioms (1)-(12) are complete with respect to the class* **NS**.

The proof of the completeness theorem is rather long and appears in the Appendix together with the other proofs. Our construction employs Henkin style witnesses (see (Enderton, 1972)) which are repeatedly added to the language in order to guarantee the existence of the non-normal objects stipulated by the theory.

Theorem 4 *For a countable language $\mathcal{L}_C$*

1. *Axioms (1)-(13) are complete w.r.t. the class of* **NS** *structures satisfying Permutation.*
2. *Axioms (1)-(17) are complete w.r.t. the class of* **NS** *structures satisfying* permutation *and* concatenation.

4 PROBABILISTIC SEMANTICS

The existence of probabilistic semantics for propositional conditional logics has been pointed out by a number of authors, for example Adams (1975), Goldszmidt and Pearl (1992), and Lehmann and Magidor (1992).

We claim that conditional first-order theories can be given probabilistic semantics similar to that of ϵ-semantics (Adams, 1975; Goldszmidt, Morris, & Pearl, 1993). The intuition behind this semantics is as follows: Suppose there is a probability distribution Pr defined over the domain of a first order model $\mathcal{M}$, and let us abuse notation and write $Pr(\varphi(x))$ for $Pr(\{d : \mathcal{M} \models \varphi(x)[^x_d]\})$. We would like to say that a conditional $\varphi \rightarrow_{(x)} \psi$ is satisfied by this model when $Pr(\psi(x)|\varphi(x))$ is almost 1. That is, with high probability, any element that has the property φ, has the property ψ. In order to formalize this intuition we have to say what we mean by "almost 1" and how we would treat tuples of domain elements, as opposed to single domain elements. We resolve the first question by using the ideas of Goldszmidt, Morris and Pearl (1993) in their formulation of ϵ-semantics. Instead of looking at a single probability assignment, we look at a sequence of probability assignments and replace the requirement $Pr(\psi(x)|\varphi(x))$

is "almost" 1 with $Pr(\psi(x)|\varphi(x)) = 1$ in the limit. (An alternative approach is to use non-standard probabilistic measures, where "almost" 1 means "within an infinitesimal distance to 1", see (Lehmann & Magidor, 1992).) The second problem is ideally resolved by using the probability distribution over the domain to induce a probability distribution over tuples, i.e, $Pr((d_1, \ldots, d_k)) = Pr(d_1) \cdots Pr(d_k)$. Unfortunately, our language is not strong enough to force such an interpretation. However, we can obtain this condition by resorting to some form of ω-consistency. We conjecture that this property of ω-consistency follows from regular consistency when the theory is finite.

Definition 8 *A parameterized probabilistic first-order model (PPF) is a pair* $\mathcal{M}_P = \langle \mathbf{M}, P \rangle$, *where*

- $\mathbf{M}$ *is a standard first-order structure.*
- $P = \{Pr^k : k \in N\}$, *where* $Pr^k = \{Pr^k_n : n \in N\}$ *is a sequence of probability measures on* $|\mathbf{M}|^k$.

Satisfiability of a formula φ in a PPF $\mathcal{M}_P$ is defined in the standard fashion, except for the following case:

- $\mathcal{M}_P \models \beta \rightarrow_{(\bar{x})} \gamma$ if $\lim_{n\to\infty} Pr_n^{|\bar{x}|}(\{\bar{c} : \mathcal{M}_P \models \gamma[\frac{\bar{x}}{\bar{c}}]\} \mid \{\bar{c} : \mathcal{M}_P \models \beta[\frac{\bar{x}}{\bar{c}}]\}) = 1$. (We abbreviate this as $\lim_{n\to\infty} Pr(\gamma|\beta) = 1$.)

That is, the conditional $\beta \rightarrow_{(\bar{x})} \gamma$ is satisfied whenever the probability of the set of substitutions for $\bar{x}$ under which $\beta \wedge \gamma$ is satisfied, given the set of substitutions under which β is satisfied, approaches 1.

Definition 9 *A PPF is* **strong** *if for all* $k, n \in N$ *and for all* $d_1, \ldots, d_k \in \|\mathcal{M}\|$, *we have that* $Pr^k_n((d_1, \ldots, d_k)) = Pr^1_n(d_1) \cdots Pr^1_n(d_k)$.

Theorem 5 *For a countable language* $\mathbf{L}$, *axioms (1)-(12) are sound and complete w.r.t. the class of PPF models.*

However, we conjecture that a much stronger result holds.

Conjecture 1 *For a finite theory* Γ, Γ *has a model in* **NS** *satisfying* permutation *and the* concatenation *iff* Γ *has a strong PPF model.*

While it is easy to define a ranked structure based on a PPF model, we have not been able to prove the other direction.

We remark that Lehmann and Magidor (1992) give a slightly different semantics to propositional conditional logics, using non-standard probability measures.

5 DISCUSSION

We presented a first-order conditional logic in which conditionals are interpreted as qualitative statistical statements. Our main contributions are in pointing out this alternative interpretation for conditionals, one that underlies much common-sense knowledge; in providing a formal language for making such statements, including a sound and complete axiomatization; and in pointing out its probabilistic semantics.

Particularly relevant to our formalism are the work of Schlechta (1995) on defaults as generalized quantifiers and the work of Friedman, Halpern, and Koller (1996) on first-order conditional logics. Schlechta interprets defaults of the form "birds normally fly" as saying that "most birds fly" or that "the elements of a large or important subset of the set of birds fly." Hence, as in our logic, defaults (or conditionals) are interpreted w.r.t. a single first-order model. Schlechta formalizes the notion of "important subset" using a weak notion of a *filter*. A filter w.r.t. a set A is a set of subsets of A that contains A, that is closed under the superset relation and under intersections. Schlechta replaces this last property with the requirement of non-empty intersections. Hence, Schlechta's semantics consists of a first-order structure supplemented by weak filters that tell us for each set what its important subsets are. Schlechta provides a sound and complete axiomatization for his logic.

Schlechta's language contains unary predicates only. Hence, it provides only a limited formalization of first-order conditionals. Consequently, it is less expressive than our logic and does not address issues such as the interaction between normal tuples of different sizes. However, these works are closely related. Schlechta's axiom system is weak and, consequently, flexible. In fact, restricted to unary predicates, our semantics is an extension of Schlechta's semantics, since rankings implement a (real) filter which, in turn, is a weak filter. Hence, we have formalized a more specific approach to conditionals. Indeed, while our conditional operator can be given probabilistic semantics as stating statements that are almost certainly true, one possible interpretation of Schlechta's defaults is as stating facts that are true with probability greater than 0.5. Our more specific semantics, originally presented in (Brafman, 1991), has the benefit of a simpler representation: whereas one must specify a filter for every subset of the domain in order to handle conditionals in Schlechta's logic, a single ranking on the domain suffices in our case. Finally, it should be mentioned that Schlechta's work is concerned with issues which we have not considered, such as specificity of defaults.

Friedmann, Halpern, and Koller (1996) discuss first-order conditional logics in the context of plausibility measures (Friedman & Halpern, 1995), which are use to assign degrees of plausibility to formulas. Plausibility measures are a generalization of probability measures and other similar formalisms, in particular, possibility measures and rankings. While they do not present an axiomatization of the statistical interpretation and are mostly concerned with the subjective case, their semantic approach to statistical conditionals is quite promising and their language is as general as ours.

Delgrande (1987) was the first to offer a first-order version of conditional logic. His language is restricted to flat formulas, i.e., formulas in which conditionals are not nested, and his semantics is subjective. His models tells us what worlds the agent perceives as more reasonable, and conditionals are interpreted as talking about the properties of the most likely worlds.

Lehmann and Magidor (1990) offer a hybrid approach to first-order conditionals, combining elements of the subjective and statistical approaches, and using a language does allows flat conditionals only. Lehmann and Magidor provide semantics for their language using a partial order over pairs of the form $\langle M, f\rangle$, where M is a standard first-order model, and f is a function assigning a domain element to each variable. The conditional $bird(x) \rightarrow fly(x)$ is satisfied if in those minimal $\langle M, f\rangle$ pairs in which the object assigned to x by f has the property *bird*, that object also has the property *fly*. Thus, the model can be viewed as saying what models are most normal, but also what assignments to objects are most normal. While the ordering over models fits the subjective approach, the fact that there is an ordering over the assignment functions f is more in line with the statistical approach: If one fixes the model M, the ordering over assignment functions can be understood as an ordering over infinite tuples of objects which is closer to the statistical approach. Motivated by their work, Brafman (1991) uses such orders over assignment functions to define the semantics of statistical conditionals.

Friedman, Halpern, and Koller (1996) discuss a number of problems that arise in the logic of Delgrande and in other formulations of first-order conditional logic, which we mentioned in the Introduction. In Delgrande's logic the following is inconsistent:

$$\forall x, y(Pet(x, y) \rightarrow (Dog(y) \vee Cat(y)),$$

$$\forall y(Pet(John, y) \rightarrow (Snake(y)).$$

This inconsistency stems from our ability to substitute *John* for the variable x in the first sentence, obtaining a sentence that says that John's typical pets are dogs or cats. This contradicts the second statement which says that John's pets are typically snakes (assuming, of course, that snakes are not dogs or cats). In order to express similar information in our language, we would write $Pet(x, y) \rightarrow_{(x,y)} (Dog(y) \vee Cat(y))$. That is, normally, a person's pet is either a dog or a cat. Because this form of quantification is weaker than universal quantification, substitutions for x or y is not allowed.

An instance of a similar problem is the lottery paradox. We know that most tickets do not win the lottery, but there is a winning ticket. The theory:

$$\forall x(true \rightarrow \neg Winner(x)) \ , \ true \rightarrow \exists x Winner(x).$$

is inconsistent in Delgrande's logic. However, if we use statistical conditionals to represent similar information we obtain the following consistent theory:

$$true \rightarrow_{(x)} \neg Winner(x) \ , \ \exists x Winner(x).$$

It is important to note that the lottery paradox still possess a challenge to the subjective approach: It is not unreasonable to argue that the following (subjective) statement is consistent: "For every given ticket, I believe that *that* ticket will not win the lottery. Yet, I believe that some ticket will win the lottery." It seems that these statements, again, correspond to the following theory:

$$\forall x(true \rightarrow \neg Winner(x)) \ , \ true \rightarrow \exists x Winner(x).$$

with conditionals interpreted subjectively. Only now, we have made this challenge more crisp because it is not conflated with statements such as "most tickets will not win the lottery." Friedman *et al.* (1996) offer one possible solution.

Our logic suffers from a number of weaknesses. As we have seen, it is not adequate for expressing subjective statements such as: for every ticket, I believe it is most likely not to win the lottery. In addition, Axiom (12) is not always desirable. As shown in (Friedman et al., 1996), a consequence of

$$\forall y(true \rightarrow_{(x)} \neg Married(x, y))$$

would be

$$true \rightarrow_{(x)} \forall y \neg Married(x, y)).$$

That is, if we assert for any individual y, that it is not married to most other individuals, we conclude that most individuals are not married. This problem does not exist when one does not require rankings to

be *smooth*, or when the semantics of (Friedman et al., 1996) is used.

The fact that we cannot substitute terms for variables bound by the conditional operator also means that we cannot conclude $fly(Tweety)$ from $bird(x) \rightarrow_{(x)} fly(x)$. That is, although we know that most/normal birds fly, we have no mechanism to deduce that Tweety is one of these normal birds. This is not surprising, since we cannot conclude that Tweety flies given a statistical statement such as "99% of all birds fly." In order to handle this we would need additional nonmonotonic machinery. For instance, one would like to deduce that an object satisfying a formula is normal for that formula unless the opposite is known, i.e., Tweety is a normal bird unless we can prove otherwise. Such deductions are not possible within our logic, which is a monotonic formalism.

The problem we just alluded to is a qualitative version of a fundamental problem at the heart of statistical reasoning: Given statistical information, what is it that we should believe about particular individuals? That is, if we know that 99% of birds fly, what should we believe about Tweety. Bacchus, Grove, Halpern and Koller (1995) proposed a solution to this problem in the quantitative case, i.e., a method for moving from statistics to beliefs, but it is not clear whether their ideas can be used to solve the qualitative version of this problem. This paper does not attempt to resolve this question, but takes a necessary first step to the resolution of this problem by making explicit the distinction between statistical and subjective conditionals and by supplying a logic for reasoning about statistical statements. We believe that a natural next step is to provide a semantically appealing logic that combines both statistical and subjective conditionals, using which the relationship between both types of information can be studied.

A PROOF OF THEOREM 3

- Lemmas and proofs marked by (LM$\star$) were lifted to the first-order case from similar proofs appearing in (Lehmann & Magidor, 1992).
- The proof of the deduction theorem trivially generalizes to our logic and is (implicitly) used in the proof.
- For a full version of this and other proofs see the author's home page.

The following derived rules will be useful:

18. $\neg(\alpha \rightarrow_{(\bar{x})} \beta) \Rightarrow \exists \bar{x}(\alpha \wedge \neg\beta)$
19. $\neg(\alpha \rightarrow_{(\bar{x})} \beta) \Rightarrow$
    $\exists \bar{x}[(\alpha \wedge \neg\beta) \wedge ((\alpha \rightarrow_{(\bar{x})} \gamma) \Rightarrow (\alpha \Rightarrow \gamma))]$
20. $\neg(\alpha \rightarrow_{(\bar{x})} \beta) \wedge (\alpha \rightarrow_{(\bar{x})} \gamma) \Rightarrow \exists \bar{x}(\alpha \wedge \gamma \wedge \neg\beta)$
21. $\exists \bar{x}\alpha(\bar{x}) \Rightarrow \alpha \rightarrow_{(\bar{x})} \forall \bar{y}(\gamma(\bar{x}, \bar{y}) \vee \neg\alpha \rightarrow_{(\bar{x})} \gamma(\bar{x}, \bar{y}))$
    $(\bar{y} \cap (\bar{x} \cup \mathrm{fv}(\alpha)) = \emptyset)$
22. $(\alpha \wedge \beta) \rightarrow_{(\bar{x})} \gamma \Rightarrow \alpha \rightarrow_{(\bar{x})} (\beta \Rightarrow \gamma)$ $\quad (LM\star)$
23. $((\alpha \vee \beta) \rightarrow_{(\bar{x})} \neg\beta) \Rightarrow (\alpha \rightarrow_{(\bar{x})} \neg\beta)$ $\quad (LM\star)$
24. $(\alpha \rightarrow_{(\bar{x})} \mathbf{false}) \Rightarrow ((\alpha \wedge \beta) \rightarrow_{(\bar{x})} \mathbf{false})$ $\quad (LM\star)$
25. $(\alpha \vee \beta \vee \gamma) \rightarrow_{(\bar{x})} (\neg\alpha \wedge \neg\beta)$
    $\Rightarrow (\beta \vee \gamma) \rightarrow_{(\bar{x})} \neg\beta$ $\quad (LM\star)$
26. $(\alpha \vee \beta) \rightarrow_{(\bar{x})} \neg\alpha \vdash (\alpha \vee \beta \vee \gamma) \rightarrow_{(\bar{x})} \neg\alpha$ $\quad (LM\star)$

We use $\hat{\beta}_{\bar{x}}$ (or $\hat{\beta}$ when $\bar{x}$ is clear from the context) to denote the set $\{\bar{c} \in |\mathbf{M}|^{|\bar{x}|} : \mathcal{M} \models \beta[\frac{\bar{x}}{\bar{c}}]\}$. We use the letters f, g to refer to tuples of domain elements (i.e., members of $|\mathbf{M}|^n$). Finally we define, $\mathcal{M} \models_f \alpha \stackrel{\text{def}}{=} \mathcal{M} \models \alpha[\frac{\bar{x}}{f}]$ (where $|f| = |\bar{x}|$).[4]

Theorem 1 *For a countable language $\mathcal{L}_C$, axioms (1)-(12) are complete with respect to the class* **NS**.

Before we start the proof, we mention that we will need a syntactic notion of substitution. We will use the same notation as in the case of the (semantic) assignment function, but which one is intended would be clear from the context. The definition will be identical to the one used in first-order logic, where basically, $\alpha[\frac{x}{c}]$ says that we must substitute the constant $c \in \mathcal{L}_C$ for all those instances of x in the wff α, where x appears free in α. So $Man(x)[\frac{x}{c}] = Man(c)$, but $Man(x) \rightarrow_{(\bar{x})} Tall(x)[\frac{x}{c}]$ is $Man(x) \rightarrow_{(\bar{x})} Tall(x)$. Thus, syntactic substitution defines a mapping from $\mathcal{L}_C$ to $\mathcal{L}_C$.

Proof: Let $\mathcal{L}_C$ be a countable language and $\hat{\Gamma}$ a consistent set of closed wffs in $\mathcal{L}_C$. We shall build a model $\mathcal{M}$ s.t. $\mathcal{M} \models \hat{\Gamma}$. We shall construct the model in ω stages. Each stage will deal with one negated conditional $\neg(\alpha \rightarrow_{(\bar{x})} \beta)$.

At each stage we will define five sets $\mathbf{L}_n, \mathbf{\Gamma}_n, \mathbf{F}_n$, $\mathbf{E}_n$ and $\mathbf{E}^n$. We initialize $\mathbf{L}_0, \mathbf{\Gamma}_0, \mathbf{F}_0$ and $\mathbf{E}_0$ as follows:

- $\mathbf{L}_0 = \mathcal{L}_C$.
- $\mathbf{\Gamma}_0 \supseteq \hat{\Gamma}$, is a maximal consistent set of closed wffs. (We remark that consistency is defined as usual and if $\mathbf{\Gamma}$ is maximal consistent, then forall $\alpha \in \mathbf{L}$ we have $\alpha \in \mathbf{\Gamma}$ or $\neg\alpha \in \mathbf{\Gamma}$).

[4] Recall that $|f|$ stands for the cardinality of $|f|$. Unfortunately, when $\mathbf{M}$ is a first-order structure, $|\mathbf{M}|$ stands for the domain of $\mathbf{M}$. However, this is the only exception.

- $\mathbf{F}_0 = \emptyset$
- $\mathbf{E}_0$ is an enumeration of pairs of wffs in $\mathbf{L}_0$ defined as $\{\langle \alpha, \beta \rangle \mid \neg(\alpha \rightarrow_{(\bar{x})} \beta) \in \mathbf{\Gamma}_0$.
- $\mathbf{E}^0 = \mathbf{E}_0$.

Definition 10 *A* **type** $\Gamma(x_1, \ldots, x_k)$ *in the variables* $x_1, \ldots, x_k$ *is a maximal consistent set of formulas of* $\mathbf{L}$ *in these variables.* [5]

Assume that we have defined $\mathbf{L}_n, \mathbf{\Gamma}_n, \mathbf{F}_n, \mathbf{E}_n$ and $\mathbf{E}^n$. Let $\langle \alpha, \beta \rangle$ be the $n+1^{th}$ pair in the enumeration $\mathbf{E}^n$. Define $\mathbf{NC}(\alpha)^k_n = \{\gamma : (\alpha \rightarrow_{(\bar{x})} \gamma) \in \mathbf{\Gamma}_n$ and $|\bar{x}| = k\}$ and let τ be a type in the variables $\bar{x}$ with respect to $\mathbf{\Gamma}_n$ *in* $\mathbf{L}_n$ (i.e., containing $\mathbf{\Gamma}_n$), such that $\mathbf{NC}(\alpha)^k_n \cup \{\neg\beta\} \subset \tau$.

Lemma 1 *Such a type* τ *exists.*

Proof: We must show that $\mathbf{\Gamma}_n \cup \{\neg\beta\} \cup \mathbf{NC}(\alpha)^k_n$ is consistent. We shall prove that
$\mathbf{\Gamma}_n \vdash \exists\bar{x}(\neg\beta \wedge \gamma_1 \wedge \ldots \wedge \gamma_m)$ for any finite subset $\{\gamma_1, \ldots, \gamma_m\} \subset \mathbf{NC}(\alpha)^k_n$.
We know that $\alpha \rightarrow_{(\bar{x})} \gamma_1, \ldots, \alpha \rightarrow_{(\bar{x})} \gamma_m \in \mathbf{\Gamma}_n$. Therefore by using **And** we obtain $\alpha \rightarrow_{(\bar{x})} (\gamma_1 \wedge \ldots \wedge \gamma_n) \in \mathbf{\Gamma}_n$ (by maximal consistency). $\langle \alpha, \beta \rangle \in \mathbf{E}_j$ for some $j \leq n \implies \neg(\alpha \rightarrow_{(\bar{x})} \beta) \in \mathbf{\Gamma}_n$ (by definition of $\mathbf{E}_j$).
By (19) and $\mathbf{\Gamma}_n$'s maximal consistency we have $\exists\bar{x}(\neg\beta \wedge \gamma_1 \wedge \ldots \wedge \gamma_m) \in \mathbf{\Gamma}_n$. ∎

We can now define the $(n+1)^{th}$ stage in our construction.

- $\mathbf{L}_{n+1} = \mathbf{L}_n \cup c^{n+1}_0, c^{n+1}_1, \ldots, c^{n+1}_{k-1}$ where $c^{n+1}_m \notin \mathbf{L}_n$ for all $m < k$.
- $\mathbf{\Gamma}_{n+1} = \mathbf{\Gamma}_n \cup \tau[^{x_0, x_1, \ldots, x_{k-1}}_{c^{n+1}_0, \ldots, c^{n+1}_{k-1}, \ldots, c^{n+1}_{k-1}}]$ where we have assumed $x_0, \ldots, x_{k-1}$ are the variables occurring in τ.
- $\mathbf{F}_{n+1} = \mathbf{F}_n \cup f^\alpha_{n+1}$ where $f^\alpha_{n+1} = (c^{n+1}_0, \ldots, c^{n+1}_{k-1})$.
- $\mathbf{E_{n+1}}$ is an arbitrary enumeration of pairs $\langle \alpha', \beta' \rangle$ s.t. $(\alpha' \wedge \beta') \in \mathbf{L}_{n+1} \setminus \mathbf{L}_n$ and $\neg(\alpha' \rightarrow_{(\bar{x})} \beta') \in \mathbf{\Gamma}_{n+1}$.
- $\mathbf{E}^{n+1}$ is the enumeration obtained by using Cantor's diagonalization method to order the elements of $\mathbf{E}_0, \ldots, \mathbf{E}_{n+1}$.

Define $\mathbf{L} = \bigcup \mathbf{L}_n$, $\mathbf{\Gamma} = \bigcup \mathbf{\Gamma}_n$, $\mathbf{F} = \bigcup \mathbf{F}_n$. It is clear that $\mathbf{\Gamma}$ is maximal consistent in $\mathbf{L}$, because each of the $\mathbf{\Gamma}_n$ is maximal consistent in $\mathbf{L}_n$ for each $n \in \mathbf{N}$ and the union itself is consistent (since $\mathbf{\Gamma}_n \subset \mathbf{\Gamma}_{n+1}$).

Lemma 2 *1. For each* $\mathbf{\Gamma}$*-consistent wff* α *we have a tuple* $f^\alpha \in \mathbf{F}$ *such that* $|f^\alpha| = |\bar{x}|$ *and for any* γ *s.t.* $\alpha \rightarrow_{(\bar{x})} \gamma \in \mathbf{\Gamma}$, *we have that* $\gamma[^{\bar{x}}_{f^\alpha}] \in \mathbf{\Gamma}$. *We shall call* f^α **normal**$_{\bar{x}}$ *for* α .

2. For each α *and* β *s.t.* $\neg(\alpha \rightarrow_{(\bar{x})} \beta) \in \mathbf{\Gamma}$, *we have some* f *that is normal*$_{\bar{x}}$ *for* α *such that* $\neg\beta[^{\bar{x}}_f] \in \mathbf{\Gamma}$.

Proof: If α is $\mathbf{\Gamma}$-consistent then $\neg(\alpha \rightarrow_{(\bar{x})} \neg\alpha) \in \mathbf{\Gamma}$, otherwise by **Reflexivity** and **And** one has $\alpha \rightarrow_{(\bar{x})}$ **false** $\in \mathbf{\Gamma} \implies \forall\bar{x}\neg\alpha \in \mathbf{\Gamma}$ contradicting α 's $\mathbf{\Gamma}$-consistency.
Therefore, it is enough to prove the second part of the lemma.
There exist some n s.t. $\alpha, \beta \in \mathbf{L}_n \setminus \mathbf{L}_{n-1}$ and due to our construction process, this means $\neg(\alpha \rightarrow_{(\bar{x})} \beta) \in \mathbf{\Gamma}_n \setminus \mathbf{\Gamma}_{n-1}$. The fact that $\neg(\alpha \rightarrow_{(\bar{x})} \beta) \in \mathbf{\Gamma}_n$ implies that $\langle \alpha, \beta \rangle \in \mathbf{E}_n$. Therefore, there exists some $m > n$ s.t. we have chosen to deal with the pair $\langle \alpha, \beta \rangle$ in the m^{th} stage.

Claim 1 f^α_m *is normal for* α *and* $\neg\beta[^{\bar{x}}_{f^\alpha_m}] \in \mathbf{\Gamma}$

Proof: By the construction process we know that $\neg\beta[^{\bar{x}}_{f^\alpha_m}] \in \mathbf{\Gamma}_m \subset \mathbf{\Gamma}$, and that for each γ s.t. $\alpha \rightarrow_{(\bar{x})} \gamma \in \mathbf{\Gamma}_{m-1}$, we have $\gamma[^{\bar{x}}_{f^\alpha_m}] \in \mathbf{\Gamma}_m \subset \mathbf{\Gamma}$. Assume δ is such that $\alpha \rightarrow_{(\bar{x})} \delta \in \mathbf{\Gamma} \setminus \mathbf{\Gamma}_m$. Therefore there exists some $k \geq m$ s.t. $\alpha \rightarrow_{(\bar{x})} \delta \in \mathbf{\Gamma}_k \setminus \mathbf{\Gamma}_{k-1}$. This means that $\delta = \delta'(\bar{x}, \bar{c})$ where $\bar{c} \in \mathbf{L}_k \setminus \mathbf{L}_m$. But we know by virtue of $\alpha[^{\bar{x}}_{f^\alpha_m}] \in \mathbf{\Gamma}_m$ that $\exists\bar{x}\alpha \in \mathbf{\Gamma}_n$, (because $\mathbf{\Gamma}_n$ is maximal consistent and $\exists\bar{x}\alpha \in \mathbf{L}_n$. By (20) we know that $(\alpha \rightarrow_{(\bar{x})} \forall\bar{y}(\delta'(\bar{x}, \bar{y}) \vee \neg(\alpha \rightarrow_{(\bar{x})} \delta'(\bar{x}, \bar{y})))) \in \mathbf{\Gamma}_n$ too. Therefore, $\forall\bar{y}(\delta'(\bar{x}, \bar{y}) \vee \neg(\alpha \rightarrow_{(\bar{x})} \delta'(\bar{x}, \bar{y})))[^{\bar{x}}_{f^\alpha_m}] \in \mathbf{\Gamma}_m$. By substitution (of FOL) and maximal consistency, we know $\delta'(\bar{x}, \bar{c}) \vee \neg(\alpha \rightarrow_{(\bar{x})} \delta'(\bar{x}, \bar{c}))[^{\bar{x}}_{f^\alpha_m}]$ is in $\mathbf{\Gamma}_m$, and by maximal consistency and the fact that $\neg(\alpha \rightarrow_{(\bar{x})} \delta'(\bar{x}, \bar{c})) \notin \mathbf{\Gamma}_k$ we know that $\delta'(\bar{x}, \bar{c})[^{\bar{x}}_{f^\alpha_m}] \in \mathbf{\Gamma}_k \subset \mathbf{\Gamma}$ ∎This proves the whole lemma. ∎

We can now define our model $\mathcal{M} \stackrel{\text{def}}{=} \langle \mathbf{M}, \mathbf{R} \rangle$.

- $\mathbf{M}$ is the standard FOL structure defined by the standard part of $\mathbf{\Gamma}$ (as in (Enderton, 1972)).
- $\mathbf{R} = \{R_n | n \in N\}$ where R_k is a ranking function over $||\mathcal{M}||^k$ that we will now construct.

Fix some integer n. We will now construct the ranking $\mathbf{R}_n$. The construction is identical for every n.

[5] This definition is from (Chang & Keisler, 1990).

Definition 11 *Let* $\mathbf{S}$ *be the set of all* Γ*-consistent wffs (both open and closed), fix some ordered set of n (different) variables* $\bar{x}$ *and let* $\mathcal{R} \in \mathbf{S} \times \mathbf{S}$ *where* $\alpha \mathcal{R} \beta$ *iff* $\neg(\alpha \vee \beta \rightarrow_{(\bar{x})} \neg\alpha) \in \Gamma$.

Lemma 3 $\mathcal{R}$ *is transitive.* $(LM*)$

Lemma 4 *For each* $\alpha, \beta \in \mathbf{S}$, *either* $\alpha \mathcal{R} \beta$ *or* $\beta \mathcal{R} \alpha$. $(LM*)$

Lemma 5 *If* $\alpha \mathcal{R} \beta$ *then for any* f *normal*$_{\bar{x}}$ *for* α *s.t.* $\beta[^{\bar{x}}_{f}] \in \Gamma$, f *is normal*$_{\bar{x}}$ *for* β . $(LM*)$

Definition 12 $\alpha \sim \beta$ *for* $\alpha, \beta \in \mathbf{S}$ *iff* $\alpha \mathcal{R} \beta$ *and* $\beta \mathcal{R} \alpha$.

The previous lemmas imply that $\sim$ is an equivalence relation. We denote the equivallence class of α by $\bar{\alpha}$, we will denote the set of equivallence classes by $\mathbf{E}$. The relationship $\mathcal{R}$ on formulas induces an ordering over equivallence classes in $\mathbf{E}$ defined below.

Definition 13

- $\bar{\alpha} \stackrel{\text{def}}{=} \{\gamma \in \mathbf{S} : \alpha \sim \gamma\}$,
- $\mathbf{E} \stackrel{\text{def}}{=} \{\bar{\alpha} : \alpha \in \mathbf{S}\}$,
- $\bar{\alpha} \leq \bar{\beta}$ *iff* $\alpha \mathcal{R} \beta$,
- $\bar{\alpha} < \bar{\beta}$ *iff* $\bar{\alpha} \leq \bar{\beta}$ *and* $\alpha \not\sim \beta$.

The previous lemmas imply that $<$ is a strict total order on $\mathbf{E}$.

Lemma 6 *If* $\alpha, \beta \in \mathbf{S}$ *and* $\bar{\beta} < \bar{\alpha}$ *then* $\beta \rightarrow_{(\bar{x})} \neg\alpha$. $(LM*)$

Lemma 7 *Let* $\alpha, \beta \in \mathbf{S}$. *If there exists some* $f \in \mathbf{F}$ *such that* f *is normal*$_{\bar{x}}$ *for* α *and* $\beta[^{\bar{x}}_{f}] \in \Gamma$, *then* $\bar{\beta} \leq \bar{\alpha}$. $(LM*)$

We can now define $\mathbf{R}$ (actually R_n). Let $|f^\alpha| = |g^\beta| = |f| = |g| = n$

- If $f^\alpha, g^\beta \in \mathbf{F}$ then $f^\alpha <_{\mathbf{R}} g^\beta$ iff $\bar{\alpha} < \bar{\beta}$.
- If $f \in \mathbf{F}$, $g \in \bar{\mathbf{F}} \setminus \mathbf{F}$ then $f <_{\mathbf{R}} g$.
- If both $f, g \in \bar{\mathbf{F}} \setminus \mathbf{F}$ then $f =_{\mathbf{R}} g$.

$\mathbf{R}$ is well defined because Lemma 7 implies that if $f^\alpha \equiv g^\beta$ (i.e. f was constructed as normal$_{\bar{x}}$ for α and g was constructed as normal$_{\bar{x}}$ for β and they are identical) then $\bar{\alpha} = \bar{\beta}$. Also, notice that Axiom (11) implies that the choice of the variables $\bar{x}$ in the definition of normal$_{\bar{x}}$ does not matter, and that different rankings obtained using different sets of variables will be identical.

Definition 14 *We say that* f satisfies$_{\bar{x}}$ α , *written* $f \models_{\bar{x}} \alpha$ *iff* $\alpha[^{\bar{x}}_{f}] \in \Gamma$. *We say that* f *is* minimal$_{\bar{x}}$ for α *iff* $f \models_{\bar{x}} \alpha$ *and* $g \models_{\bar{x}} \alpha$ *implies that* $f \leq_{\mathbf{R}} g$. *When* $\bar{x}$ *is clear from the context we will simply say that* f *satisfied* α *or that* f *is minimal for* α *and write* $f \models \alpha$.

Lemma 8 *If* f *is minimal for* α, *then* $f \in F$. $(LM*)$

We conclude that any minimal function is normal for some wff.

Lemma 9 *Let* $f \in \mathbf{F}$ *be normal for* α. *Then* f *is minimal in* $\hat{\beta}$ *iff* $f \models \beta$ *and* $\bar{\beta} = \bar{\alpha}$. $(LM*)$

Lemma 10 *Let* $\alpha \in \mathbf{S}$. $\hat{\alpha}$ *has a minimal element.* $(LM*)$

Lemma 11 *If* $g \in \mathbf{F}$ *is normal for* α *and minimal in* $\hat{\beta}$, *then* g *is normal for* β . $(LM*)$

Note that $g \in \mathbf{F} \implies \exists \alpha \in \mathbf{S}$ s.t. g is normal for α . Because Lemma 9 tells us that if g is minimal for α it is in F, we can conclude that if g is minimal for α then g is normal for α.

Proof of completeness theorem: We now show that $\mathcal{M} \models \Gamma$. Since our original set of wffs $\hat{\Gamma} \subset \Gamma$, this is enough. By induction on the structure of $\alpha \in \Gamma$ we show that $\mathcal{M} \models \alpha \Leftrightarrow \alpha \in \Gamma$. We have implicitly assumed that $\{=\} \notin \mathbf{L}$. The generalization to languages containing equality proceeds as in the Henkin's proof for standard first-order logic. (see, e.g., (Enderton, 1972)). Our construction augments Henkin's proof to deal with the conditional operator, and the standard steps are conducted as in Henkin's proof.

- α is an atomic formula, $\alpha = \neg\beta$, or $\alpha = (\beta \Rightarrow \gamma)$. Same as in Henkin's completeness proof for FOL (e.g., (Enderton, 1972) (pp. 132-133)).
- $\alpha = \forall x \beta$. To follow Henkin's proof we need to extend the substitution lemma to our language, and use the following lemma:

  Lemma 12 *For each wff* $\varphi(x) \in \mathbf{L}$ *we have some* $c \in \mathbf{L}$ *s.t.* $\neg\forall x \varphi \rightarrow \neg\varphi[^{x}_{c}] \in \Gamma$.

  The generalization of the substitution lemma is straigtforward since the conditional operator behaves much like the universal quantifier.
- $\alpha = (\beta \rightarrow_{(\bar{x})} \gamma)$. First, assume that $\beta \rightarrow_{(\bar{x})} \gamma \in \Gamma$. if $\beta \in \mathbf{S}$ we know by Lemma 1 that $\exists f \in \mathbf{F}$ normal for β . By Lemma 11 we know that for any $g \in \mathbf{F}$ minimal in $\hat{\beta}$, it is the case that g is normal for

β . Hence, for all $g \in \hat{\beta}$ that is minimal in $\hat{\beta}$, we have that $\gamma[\frac{\bar{x}}{g}] \in \Gamma$. By the inductive hypothesis $\mathcal{M} \models \gamma[\frac{\bar{x}}{g}][\hat{s}]$. We conclude that $\mathcal{M} \models \beta \rightarrow_{(\bar{x})} \gamma$. If $\beta \notin \mathbf{S}$ then $\mathcal{M} \models \beta \rightarrow_{(\bar{x})} \gamma$ trivially.

Assume now that $\beta \rightarrow_{(\bar{x})} \gamma \notin \Gamma$. By maximal consistency $\neg(\beta \rightarrow_{(\bar{x})} \gamma) \in \Gamma$. This implies (by (18)) that $\beta \in \mathbf{S}$. Lemma 2 shows that there exists $f \in \mathbf{F}$ normal for β, s.t. $\neg\gamma[\frac{\bar{x}}{f}] \in \Gamma$. Lemma 9 implies that f is minimal in $\hat{\beta}$. This implies (by using the inductive hypothesis to show that $\mathcal{M} \models \neg\gamma[\frac{\bar{x}}{f}][\hat{s}]$) that $\mathcal{M} \not\models \beta \rightarrow_{(\bar{x})} \gamma[\hat{s}]$. $\Longrightarrow \mathcal{M} \models \neg(\beta \rightarrow_{(\bar{x})} \gamma)g[\hat{s}]$.

Finally, we note that in order to imitate Henkin's proof we need to establish the semantic and syntactic equivalence of alphabetic variants. The syntactic equivalence is proven using Axiom (11) (**Renaming**) for the case of normally quantified wffs. The semantical equivalence follows from soundness. ∎

Acknowledgments

This work is based on (Brafman, 1991), conducted under the guidance of Menachem Magidor, whose technical expertise, time, and support were essential for its success. Numerous important comments and suggestions regarding that work were provided by Daniel Lehmann. This version owes much of its existence to the encouragement of Nir Friedman, who also supplied many important comments during numerous discussions. Finally, Joe Halpern provided important comments on previous drafts of this paper.

References

Adams, E. (1975). *The Logic of Conditionals.* D. Reidel, Dordrecht, Netherlands.

Bacchus, F., Grove, A. J., Halpern, J. Y., & Koller, D. (1995). Statistical foundations for default reasoning. In *Proc. Thirteenth International Joint Conference on AI*, pp. 563–569.

Bacchus, F. (1990). *Representing and Reasoning With Probabilistic Knowledge.* MIT-Press, Cambridge, Massachusetts.

Boutilier, C. (1994). Conditional logics of normality: a modal approach. *Artificial Intelligence, 68*, 87–154.

Brafman, R. I. (1991). A logic of normality: Predicate calculus incorporating assertions. Master's thesis, Hebrew University of Jerusalem.

Chang, C. C., & Keisler, H. J. (1990). *Model Theory.* North Holland, Amsterdam.

Delgrande, J. P. (1987). A first-order logic for prototypical properties. *Artificial Intelligence, 33*(1), 105–130.

Enderton, H. B. (1972). *A Mathematical Introduction to Logic.* Academic Press, New York.

Friedman, N., & Halpern, J. (1995). Plausibility measures: A user's guide. In *Uncertainty in AI, Proceedings of the Eleventh Conference.*

Friedman, N., Halpern, J. Y., & Koller, D. (1996). Conditional first-order logic revisited. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence.*

Goldszmidt, M., Morris, P., & Pearl, J. (1993). A maximum entropy approach to nonmonotonic reasoning. *IEEE Transactions of Pattern Analysis and Machine Intelligence, 15*(3), 220–232.

Goldszmidt, M., & Pearl, J. (1992). Rank-based systems: A simple approach to belief revision, belief update and reasoning about evidence and actions. In *Principles of Knowledge Representation and Reasoning: Proc. Third Intl. Conf. (KR '92)*, pp. 661–672.

Halpern, J. (1990). An analysis of first-order logics of probability. *Artificial Intelligence, 46*, 311–350.

Kraus, S., Lehmann, D., & Magidor, M. (1990). Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence, 44*(1–2), 167–207.

Lehmann, D., & Magidor, M. (1992). What does a conditional knowledge base entail?. *Artificial Intelligence, 55*, 1–60.

Lehmann, D., & Magidor, M. (1990). Preferential logics: the predicate calculus case. In *Proceedings of the Third Conf. on Theoretical Aspects of Reasoning About Knowledge*, pp. 57–72 San Francisco, California. Morgan Kaufmann.

Lewis, D. K. (1973). *Counterfactuals.* Harvard University Press, Cambridge, MA.

Schlechta, K. (1995). Defaults as generalized quantifiers. *Journal of Logic and Computation, 5*(4), 473–494.

Belief Revision

Towards a Practical Approach to Belief Revision: Reason-Based Change

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

Abstract

Belief revision involves modeling changes to repositories of information. The AGM paradigm is a standard theoretical framework that supports belief revision. Its change functions can be constructed from an entrenchment ranking; the ranking specifies what information is retained and lost when new information is acquired. The iteration of change functions can be achieved by transmuting the underlying entrenchment ranking.

In this paper we describe a new reason-based transmutation known as *maxi-adjustment*. It changes an entrenchment ranking under *maximal inertia*. Using the established connection between Spohnian reasons and transmutations, we argue that maxi-adjustments are a step towards a practical approach to belief revision because they assume by default that information is *independent* with respect to change, unless dependence is derivable from the explicitly specified dependencies.

In real world applications not only is it often easier to specify dependence rather than independence, but the specification usually requires less explicit information whenever there are fewer dependencies than independencies. This situation is analogous to database applications in which there are far fewer positive facts than negative facts, and by using the closed world assumption only the positive facts need be explicitly represented.

1 INTRODUCTION

An intelligent information system must possess the ability to revise its knowledge base when it receives new information. The AGM paradigm was developed by Alchourrón, Gärdenfors and Makinson (1985), and has become one of the standard frameworks for modeling change. It provides formal mechanisms for modeling the rational evolution of an ideal corpus of information. The logical properties of a body of information are not strong enough to uniquely determine change functions, however Gärdenfors and Makinson (1988) showed that an epistemic entrenchment ordering can uniquely determine how the system will react to the intrusion of new information.

The AGM paradigm is a simple and elegant framework for modeling change, however from a practical perspective it suffers from two shortcomings. First an epistemic entrenchment ordering is not propagated by its change functions, and second an epistemic entrenchment ordering ranks the elements of a logical theory; a possibly infinite set.

The first problem has been addressed by Spohn (1988), Nayak (1994), Boutilier (1993), Lehmann (1995), Darwiche and Pearl (1994), Hansson (1989) and Williams (1994a). Solutions to the second problem have been provided by Hansson (1992), Fuhrmann (1991), Rott (1991), Makinson (1987), Nebel (1989, 1991), and Williams (1994b). Similar work in the possibilistic logic setting has been conducted by Dubois and Prade (1994), and in the probabilistic setting by Pearl (1994).

We refer to the process of revising an entrenchment ranking as a *transmutation*. Appealing to the *Principle of Minimal Change* results in two types of transmuta-

tion; conditionalization (Spohn, 1988) and adjustment (Williams, 1994a). Conditionalization is based on a *relative* measure of minimal change, whilst adjustment is based on an *absolute* measure of minimal change. A *finite partial entrenchment ranking* (Williams, 1995a) is a finite representation of an entrenchment ranking. A computational model for adjusting partial entrenchment rankings was described and analysed in (Williams, 1994a), and an object-oriented implementation was described in (Williams *et al* 1995a). Unfortunately, this procedure is awkward to use for applications in which most information is independent.

In this paper we provide a new approach to propagating a ranking under *maximal inertia*. We refer to this method as a *maxi-adjustment*. It departs substantially from the standard Gärdenfors and Makinson (1988) entrenchment construction. However, it still satisfies the basic AGM postulates (with the exception of recovery), and under certain conditions the extended ones as well.

Maxi-adjustments are motivated by the results in (Williams *et al*, 1995a) which established that Spohnian reasons (Spohn, 1983) can be captured in a natural way by minimally changing an entrenchment ranking. Maxi-adjustments assume by default that information is independent unless dependence is derivable from the explicitly specified dependencies where dependencies are represented using Spohian reasons.

The major advantage of maxi-adjustments over earlier methods is that rather than the system designer having to specify the independence of information, he specifies the dependence of information and the maxi-adjustment procedure makes the *default assumption* that information is independent whenever dependence is not derivable. In other words, maxi-adjustments use a closed world assumption with respect to information dependence under change. We argue that this is a practical solution to the problem of information conservation.

Section 2 contains some relevant background information from which we draw; we briefly describe the AGM paradigm, partial entrenchment rankings, and an existing procedure for reranking information. Section 3 describes a problem related to the specification of a partial entrenchment ranking in applications where most information is independent. Section 4 provides a new procedure, namely a maxi-adjustment, which addresses this specification problem. Section 5 illustrates the main ideas using a simple application, and in section 6 we summarize our work.

We begin with some technical preliminaries. Let $\mathcal{L}$ denote a language which contains a complete set of Boolean connectives. We will denote sentences in $\mathcal{L}$ by lower case Greek letters. We assume $\mathcal{L}$ is governed by a logic that is identified with its consequence relation $\vdash$ which is assumed to satisfy the following conditions (Gärdenfors, 1988): (a) if α is a truth-functional tautology, then $\vdash \alpha$, (b) if $\vdash (\alpha \rightarrow \beta)$ and $\vdash \alpha$, then $\vdash \beta$ (*modus ponens*), (c) $\vdash$ is consistent, i.e. $\nvdash \bot$, where $\bot$ denotes the inconsistent theory, (d) $\vdash$ satisfies the deduction theorem, and (e) $\vdash$ is compact.

The set of all logical consequences of a set $T \subseteq \mathcal{L}$, i.e. $\{\alpha : T \vdash \alpha\}$, is denoted by $\mathrm{Cn}(T)$. A *theory* of $\mathcal{L}$ is any subset of $\mathcal{L}$ closed under Cn. We let $\mathcal{L}^{\bowtie}$ denote the set of *contingent* sentences. Finally, a *well-ranked* preorder on a set Γ is a preorder such that every nonempty subset of Γ has a minimal member.

2 BELIEF REVISION

2.1 The Rationality Postulates

Within the AGM paradigm a body of information is represented as a theory, and informational changes are regarded as transformations on theories. The principal types of AGM transformations are expansion, contraction, withdrawal, and revision. These transformations allow us to model changes of information based on the *Principle of Minimal Change*. Expansion is the simplest change, it models the acquiescence of information without the removal of any inconsistent information. More formally, the *expansion* of a theory T with respect to a sentence α, denoted as T_α^+, is defined to be the logical closure of T and α, i.e. is $T_\alpha^+ = \mathrm{Cn}(T \cup \{\alpha\})$.

A *revision* attempts to transform a theory in order to incorporate a sentence so that the resultant theory is consistent. Formally, a *revision function* * is any function which satisfies the postulates (^{*1}) – (^{*8}), below.

(^{*1}) T_α^* is a theory
(^{*2}) $\alpha \in T_\alpha^*$
(^{*3}) $T_\alpha^* \subseteq T_\alpha^+$
(^{*4}) If $\neg\alpha \notin T$ then $T_\alpha^+ \subseteq T_\alpha^*$
(^{*5}) $T_\alpha^* = \bot$ if and only if $\vdash \neg\alpha$
(^{*6}) If $\vdash \alpha \equiv \beta$ then $T_\alpha^* = T_\beta^*$
(^{*7}) $T_{\alpha\wedge\beta}^* \subseteq (T_\alpha^*)_\beta^+$
(^{*8}) If $\neg\beta \notin T_\alpha^*$ then $(T_\alpha^*)_\beta^+ \subseteq T_{\alpha\wedge\beta}^*$

A *contraction* of T with respect to α, T_α^-, involves the removal of a set of sentences from T so that α is no longer implied. Formally, a *contraction function* $^-$ is any function which satisfies the postulates ($^-1$) – ($^-8$), below.

($^-1$) T_α^- is a theory.
($^-2$) $T_\alpha^- \subseteq T$
($^-3$) If $\alpha \notin T$ then $T_\alpha^- = T$
($^-4$) If $\nvdash \alpha$ then $\alpha \notin T_\alpha^-$
($^-5$) $T \subseteq (T_\alpha^-)_\alpha^+$ (recovery)
($^-6$) If $\vdash \alpha \equiv \beta$ then $T_\alpha^- = T_\beta^-$

($^-7$) $T^-_\alpha \cap T^-_\beta \subseteq T^-_{\alpha\wedge\beta}$

($^-8$) If $\alpha \notin T^-_{\alpha\wedge\beta}$ then $T^-_{\alpha\wedge\beta} \subseteq T^-_\alpha$

A *withdrawal* function (Makinson, 1987) is a generalized contraction function in that it satisfies all but the *recovery* postulate, ($^-5$).

2.2 Partial Entrenchment Rankings

Computer-based intelligent information systems do not explicitly represent an infinite amount of information, instead they store a finite set of information and determine what *follows* from it using various forms of reasoning; deductive, abductive, inductive, nonmonotonic, hypothetical etc.

A partial entrenchment ranking (defined below) represents a *well-ranked* epistemic entrenchment ordering of $\mathcal{L}$ (Gärdenfors and Makinson, 1988) where sentences are mapped to ordinals. The ordinal assignment can be used and viewed in two ways: (i) qualitatively; as a specification of a relative ordering of sentences, or (ii) quantitatively; as an assignment of intrinsic utility for each sentence. For a discussion of various assignment scales see Gärdenfors and Makinson (1994).

The higher the ordinal assigned to a sentence by an entrenchment ranking the more firmly held it is. Throughout the remainder of this paper it will be understood that $\mathcal{O}$ is an ordinal chosen to be sufficiently large for the purpose of the discussion.

Definition: *A* partial entrenchment ranking *is a function* $\mathbf{B}$ *from a subset of sentences in* $\mathcal{L}$ *into* $\mathcal{O}$ *such that the following conditions are satisfied for all* $\alpha \in \mathrm{dom}(\mathbf{B})$*:*

(PER1) $\{\beta \in \mathrm{dom}(\mathbf{B}) : \mathbf{B}(\alpha) < \mathbf{B}(\beta)\} \not\vdash \alpha$.

(PER2) If $\vdash \neg\alpha$ *then* $\mathbf{B}(\alpha) = 0$.

(PER3) $\mathbf{B}(\alpha) = \mathcal{O}$ *if and only if* $\vdash \alpha$.

According to this definition the sentences which are assigned ordinals higher than that assigned to a sentence α cannot entail α, inconsistent information is assigned zero, and the tautologies are assigned the highest rank. The set of all partial entrenchment rankings is denoted $\mathcal{B}$.

We refer to $\mathbf{B}(\alpha)$ as the degree of acceptance of α. The intended interpretation of a partial entrenchment ranking is that sentences assigned a degree of acceptance greater than zero form the *explicit* beliefs of the system, and their logical closure form the system's *implicit* beliefs.

Definition: *Define the* explicit information content *of* $\mathbf{B} \in \mathcal{B}$ *to be* $\{\alpha \in \mathrm{dom}(\mathbf{B}) : \mathbf{B}(\alpha) > 0\}$*, and denote it by* $\exp(\mathbf{B})$*. Similiarly, define the* implicit information content *of* $\mathbf{B} \in \mathcal{B}$ *to be* $\mathrm{Cn}(\exp(\mathbf{B}))$*, and denote it by* $\mathrm{content}(\mathbf{B})$.

We are not only interested in the degree of acceptance of explicit information, but in the degree of acceptance of sentences they entail as well.

Intuitively, a partial entrenchment ranking represents a system's incomplete preferences from which a complete entrenchment ranking can be generated. Of course there could be an infinite number of entrenchment rankings compatible with a given partial specification. The following function 'degree' derives the minimum possible degree of acceptance for implicit information as specified by a partial entrenchment ranking, and is readily implemented using a standard theorem prover (Williams *et al*, 1995b).

Definition: *Let* α *be a nontautological sentence. If* $\mathbf{B}$ *is a finite partial entrenchment ranking, then define* $\mathrm{degree}(\mathbf{B}, \alpha)$ *to be*

$$\begin{cases} \textit{largest } j \textit{ such that } \{\beta \in \exp(\mathbf{B}) : \mathbf{B}(\beta) \geq j\} \vdash \alpha \\ \qquad\qquad \textit{if } \alpha \in \mathrm{content}(\mathbf{B}) \\ 0 \qquad\qquad \textit{otherwise} \end{cases}$$

The function 'degree' can be extended to the infinite case, we give the finite case definition above for the purpose of perspicuity.

2.3 Adjustments

The informational input for contracting and revising the content of partial entrenchment rankings is a sentence alone. In contrast, when arbitrarily transmuting a partial entrenchment ranking itself, we require both a sentence and an ordinal as input; the ordinal indicating the desired new rank of the sentence. We define an adjustment for partial entrenchment rankings below (Williams, 1995a). When $\mathbf{B}$ is finite then its definition constitutes a computational model which, loosely speaking, involves successive calls to the function 'degree' for each sentence in the domain of $\mathbf{B}$.

Definition: *Let* $\mathbf{B} \in \mathcal{B}$, $\alpha \in \mathcal{L}^{\bowtie}$, *and* $i < \mathcal{O}$. *An* adjustment $\mathbf{B}^\star(\alpha, i)$ *is defined below.*

$$\mathbf{B}^\star(\alpha, i) = \begin{cases} (\mathbf{B}^-(\alpha, i)) & \textit{if } i \leq \mathrm{degree}(\mathbf{B}, \alpha) \\ (\mathbf{B}^-(\neg\alpha, 0))^+(\alpha, i) & \textit{otherwise} \end{cases}$$

where $\mathbf{B}^-(\alpha, i)(\beta)$

$$= \begin{cases} i & \textit{if } \mathrm{degree}(\mathbf{B}, \alpha) = \mathrm{degree}(\mathbf{B}, \alpha \vee \beta) \textit{ and } \mathbf{B}(\beta) > i \\ \mathbf{B}(\beta) & \textit{otherwise} \end{cases}$$

for all $\beta \in \mathrm{dom}(\mathbf{B})$, *and* $\mathbf{B}^+(\alpha, i)(\beta)$

$$= \begin{cases} \mathbf{B}(\beta) & \textit{if } \mathbf{B}(\beta) > i \\ i & \textit{if } \alpha \equiv \beta \textit{ or } \mathbf{B}(\beta) \leq i < \mathrm{degree}(\mathbf{B}, \alpha \rightarrow \beta) \\ \mathrm{degree}(\mathbf{B}, \alpha \rightarrow \beta) & \textit{otherwise} \end{cases}$$

for all $\beta \in \mathrm{dom}(\mathbf{B} \cup \{\alpha\})$.

Using the results of Gärdenfors and Makinson (1988), Williams (1995a) showed that $\mathbf{B}^*(\alpha, i)$ is a partial entrenchment ranking where α is assigned the degree of acceptance i, and if i is greater than zero then content$(\mathbf{B}^*(\alpha, i))$ is the revision function $(\text{content}(\mathbf{B}))^*_\alpha$ constructed in the obvious way from the epistemic entrenchment ordering derived from the relative ordering given by $\mathbf{B}$ using the function 'degree'. Similarly, content$(\mathbf{B}^*(\alpha, 0))$ was shown to be the withdrawal function $(\text{content}(\mathbf{B}))^-_\alpha$.

We say β *is dependent on* α in $\mathbf{B}$ if α and β are in $\mathbf{B}$, and β is assigned zero when α is assigned zero during a transmutation. In other words, β is dependent on α if β is not retained in the withdrawal of α, e.g. in the case of an adjustment β *is dependent on* α if $\alpha, \beta \in \text{content}(\mathbf{B})$ and $\mathbf{B}^*(\alpha, 0)(\beta) = 0$.

Adjustments change (or disturb) a partial entrenchment ranking in an *absolute* minimal way, i.e. the rank of each sentence changes as little as necessary to achieve the goal of assigning α the rank i (Williams, 1994). This property was exploited in (Williams *et al*, 1995a) to model Spohn's notion of *reason*, and is discussed further in section 4.1.

3 THE SPECIFICATION PROBLEM

Adjustments use a partial entrenchment ranking and the function 'degree' to model belief change for a limited reasoner based on Gärdenfors and Makinson's construction: $\beta \in \text{content}(\mathbf{B}^*(\alpha, 0))$ if and only if $\beta \in \text{content}(\mathbf{B})$ and $\text{degree}(\mathbf{B}, \alpha) < \text{degree}(\mathbf{B}, \alpha \vee \beta)$.

An impediment to using an adjustment for real world applications is that the standard entrenchment construction together with the function 'degree' results in more information being lost than is often intended. Since the procedure assumes that information is dependent if it is not explicitly independent! This means that the system designer must specify independence which for many applications is more demanding (and less natural) than specifying dependence.

Moreover, common sense seems to decree that if you believe the only reason your coffee cup leaks is because it has a hole, then when upon closer inspection you discover it does not leak at all, it would seem a rational response to retract *your cup has a leak* and *your cup has a hole*. Contrariwise, if you are unaware that holes are the reason for leaks, then you could quite happily continue to believe *your cup has a hole* even when you discover that it does not leak. In other words, common sense suggests that information should be retracted only if there is good reason to do so.

Consider the agent whose explicit beliefs are $\{\alpha, \beta\}$. Let us, for the sake of the argument, assume he believes α and β with equal strength, e.g. $\mathbf{B}(\alpha) = 2$ and $\mathbf{B}(\beta) = 2$. Using the adjustment $\mathbf{B}^*(\alpha, 0)$ to remove α results in the loss of belief in β, i.e. contracting α from $\{\alpha, \beta\}$ results in $\{\}$.

Belief in β is lost because we did not explicitly specify the independence of β on α, i.e. $\text{degree}(\mathbf{B}, \alpha) < \text{degree}(\mathbf{B}, \alpha \vee \beta)$. In fact any explicit belief ranked equally or less than α would have been removed if we neglected to specify its independence of α.

It turns out that for many applications specifying independence would be too demanding. Instead it would be more reasonable to specify dependencies, and then derive independence. So without knowing that β a reason for α, we should try and keep β when we contract α.

We have two means to achieve this goal: (i) modify the standard entrenchment construction, or (ii) use some other method for calculating the degree of acceptance of implicit information.

The function 'degree' is based on the idea that implicit information be assigned the least rank possible. Assigning a higher rank than the minimum is difficult to justify, so we pursue the first option in the following section using Spohn's notion of *reason* to capture the property of dependence. In particular, information is retained unless we can derive that it is a Spohnian reason for the information to be contracted. This results in a construction that removes information only if it absolutely must to maintain the ranking properties *(PER1) - (PER3)*.

4 A REASON-BASED SOLUTION

4.1 Maximal Information Inertia

Spohn (1983) gave the following interpretation of β being a reason for α:

> β is a *reason* for α iff raising the epistemic rank of β would raise the epistemic rank of α.

Determination of this condition is dependent on the type of transmutation employed. An adjustment performs an *absolute minimal change* to the ranking therefore it is appropriate to use it to evaluate reasons; when the epistemic rank of α is increased it seems sensible to require that this change disturb the agent's background preference ranking as little as possible.

Using the results of Williams *et al.* (1995a) we can show that for $\alpha, \beta \in \text{content}(\mathbf{B})$ if an adjustment is used to determine Spohnian reasons then β is a *reason* for α if and only if $\text{degree}(\mathbf{B}, \alpha) < \text{degree}(\mathbf{B}, \beta \rightarrow \alpha)$. Our solution to the specification problem is founded on this observation.

Gärdenfors' (1992) interpretation of reasons is: β is a reason for α if and only if α is not retained in the contraction of β, i.e. $\text{degree}(\mathbf{B}, \beta) = \text{degree}(\mathbf{B}, \alpha \vee \beta)$

according to the standard entrenchment construction assuming the domain of $\mathbf{B}$ is a theoryx. However, he identifies two problems with this interpretation; namely difficulties with the circularity of reasons arise, and multiple reasons for β cannot be modeled satisfactorily. Hence, dependence defined via the standard entrenchment construction for contraction cannot be used to capture Spohnian reasons. Note, the definition based on adjustments does not suffer from either of these problems.

Information inertia in the current context refers to the *amount* of information retained during partial entrenchment ranking transitions. *Maximal information inertia* means retaining as much explicit information as possible.

Maximal information inertia is particularly pertinent to modeling the dynamics of nonmonotonic reasoning. According to Boutilier (1993), Pearl (1994), Freund and Lehmann (1994) default information is usually quite stable. For instance, although our knowledge about the flight coefficients of particular birds may change dramatically, the default *birds normally fly* will invariably remain unchanged, therefore changes should maintain as much default information as possible.

In many applications where independencies outnumber the dependencies it would make sense to assume that information is independent if it is not explicitly dependent.

The connection between Spohnian reasons and adjustments allows us to design a method for achieving this practical approach to belief revision. In particular, all the dependences can be specified using reasons and the process of adjustment can be modified so that derivable dependences are used to retract beliefs.

Intuitively, if α and β are explicit beliefs where β is a reason for α and α is a not reason for β, then assigning α to zero should involve the lowering of β to zero as well, however lowering β to zero should not effect the assignment of α. We note that it is not a foundational approach because α could remain a belief even though all the reasons for it have been removed. Foundational behaviour, if desired, can be achieved in other ways (Fuhrmann, 1991).

4.3 Maxi-adjustments

A maxi-adjustment, defined below, modifies a partial entrenchment ranking based on the idea of *maximal inertia*. In particular, it involves changing a partial entrenchment ranking so that a sentence is assigned a new rank in such a way that the assignment of other sentences is maintained unless there is a reason to change them. Therefore as much information as possible is retained using the default assumption that information is independent unless we can derive it is dependent.

The maxi-adjustment of a finite ranking $\mathbf{B}$ is suitable for changes to finite knowledge bases where the systems designer has specified dependencies. Independence can then be assumed by default. It involves successive calls to the function 'degree' for each sentence in the domain of $\mathbf{B}$. In the next section we illustrate maxi-adjustment with a simple example, and contrast its behaviour with ordinary adjustment.

Definition: *Let $\mathbf{B} \in \mathcal{B}$ be finite. We enumerate the range of $\mathbf{B}$ in ascending order as $j_0, j_1, \ldots, j_{\mathcal{O}}$. Let $\alpha \in \mathcal{L}^{\bowtie}$, $m = \text{degree}(\mathbf{B}, \alpha)$ and $0 \leq i < \mathcal{O}$. We define the (α, i)*-maxi-adjustment of $\mathbf{B}$ *to be given by $\mathbf{B}^{\star}(\alpha, i)$.*

$$\mathbf{B}^{\star}(\alpha, i) = \begin{cases} \mathbf{B}^{-}(\alpha, i) & \text{if } i \leq \text{degree}(\mathbf{B}, \alpha) \\ (\mathbf{B}^{-}(\neg\alpha, 0))^{+}(\alpha, i) & \text{otherwise} \end{cases}$$

where we define $\mathbf{B}^{-}(\alpha, i)(\beta)$ for $\beta \in dom(\mathbf{B}) \cup \{\alpha\}$ as follows:
For β with $\mathbf{B}(\beta) > j_m$ we have $\mathbf{B}^{-}(\alpha, i)(\beta) = \mathbf{B}(\beta)$.
For β with $\mathbf{B}(\beta) = j_m$ if $\beta \notin \Gamma$ where Γ is a minimal subset of $\{\gamma : \alpha \not\vdash \gamma, \mathbf{B}(\gamma) = j_m\}$ such that $\{\delta : \mathbf{B}^{-}(\alpha, i)(\delta) > j_m\} \cup \Gamma \vdash \alpha$ then $\mathbf{B}^{-}(\alpha, i)(\beta) = \mathbf{B}(\beta)$, if $\alpha \not\vdash \beta$ and $\beta \in \Gamma$ then $\mathbf{B}^{-}(\alpha, i)(\beta) = i$, otherwise $\beta \notin dom(\mathbf{B}^{-}(\alpha, i))$.
For β with $i < \mathbf{B}(\beta) < j_m$, suppose we have defined $\mathbf{B}^{-}(\alpha, i)(\beta)$ for β with $\mathbf{B}(\beta) \geq j_{m-k}$ for $k = 0, 1, 2, \ldots, n-1$, then for β with $\mathbf{B}(\beta) = j_{m-n}$ we have $\mathbf{B}^{-}(\alpha, i)(\beta) =$

$$\begin{cases} i & \text{if either } \alpha \equiv \beta, \text{ or } \beta \in \Gamma \\ & \quad \text{where } \Gamma \text{ is a minimal subset of} \\ & \quad \{\gamma : \mathbf{B}(\gamma) = j_{m-n}\} \text{ such that} \\ & \quad \{\delta : \mathbf{B}^{-}(\alpha, i)(\delta) > j_{m-n}\} \cup \Gamma \vdash \alpha \\ \mathbf{B}(\beta) & \text{otherwise} \end{cases}$$

For β with $\mathbf{B}(\beta) \leq i$ we have $\mathbf{B}^{-}(\alpha, i)(\beta) = \mathbf{B}(\beta)$.

$\mathbf{B}^{+}(\alpha, i)(\beta) =$

$$\begin{cases} \mathbf{B}(\beta) & \text{if } \mathbf{B}(\beta) > i \\ i & \text{if } \alpha \equiv \beta \text{ or } \mathbf{B}(\beta) \leq i < \text{degree}(\mathbf{B}, \alpha \rightarrow \beta) \\ \text{degree}(\mathbf{B}, \alpha \rightarrow \beta) & \text{otherwise} \end{cases}$$

for all $\beta \in dom(\mathbf{B}) \cup \{\alpha\}$.

The following observation is based on the fact that if $\alpha, \beta \in \text{content}(\mathbf{B})$ then raising the rank of β relies on the $^{+}$ operator which is identical in both the definition of an adjustment and in the definition of a maxi-adjustment.

Observation 1: Let $\alpha, \beta \in \text{content}(\mathbf{B})$. Then β is a reason for α using an adjustment if and only if β is a reason for α using a maxi-adjustment.

Theorems 2 and 3, below, are reassuring. Theorem 2 states that maxi-adjustments of partial entrenchment rankings satisfy the first six rationality postulates for revision and the first five for withdrawal; they, like most theory *base* change mechanisms do not satisfy the recovery postulate for contraction. Nebel (1991) describes a simple method one can use to satisfy recovery which can be incorporated into a maxi-adjustment if desired. Theorem 3 says that if the set of sentences at each rank are logically irredundant then maxi-adjustments satisfy all eight postulates. In practice intelligent databases and knowledge bases typically satisfy this condition, e.g. the application described in the next section does.

Theorem 2: Let $\mathbf{B}$ be a partial entrenchment ranking. Let α be a contingent sentence. Let $\mathbf{B}^\star(\alpha, i)$ be a maxi-adjustment. Let $(\text{content}(\mathbf{B}))^\star_\alpha = \text{content}(\mathbf{B}^\star(\alpha, i))$ for any $i > 0$. Let $(\text{content}(\mathbf{B}))^-_\alpha = \text{content}(\mathbf{B}^\star(\alpha, 0))$. Then (i) * satisifes (*1) - (*6),
(ii) $^-$ satisfies($^-$1) - ($^-$4) and ($^-$6), and
(iii) If $\text{exp}(\mathbf{B})$ is a theory then $^-$ satisfies($^-$1) - ($^-$6).

Theorem 3: Let $\mathbf{B}$ be a partial entrenchment ranking. Let α be a contingent sentence. Let $\mathbf{B}^\star(\alpha, i)$ be a maxi-adjustment. Let $(\text{content}(\mathbf{B}))^\star_\alpha = \text{content}(\mathbf{B}^\star(\alpha, i))$ for any $i > 0$. Let $(\text{content}(\mathbf{B}))^-_\alpha = \text{content}(\mathbf{B}^\star(\alpha, 0))$. If for all $\beta \in \text{exp}(\mathbf{B})$ such that $\alpha \vdash \beta$ and $\alpha \equiv \beta$ we have $\mathbf{B}(\beta) \neq \mathbf{B}(\alpha)$, then * is a revision function, and $^-$ is a withdrawal function.

Maxi-adjustments aim to use a closed world assumption with respect to dependence, namely if α is a reason for β cannot be derived from the partial entrenchment ranking, then it assumes that β is independent of α so long as *(PER1) - (PER3)* are not violated.

In order to achieve this the designer specifies reasons; if β is a *reason* for α then $\text{degree}(\mathbf{B}, \beta \rightarrow \alpha)$ must be strictly larger than $\text{degree}(\mathbf{B}, \alpha)$. Specifying reasons essentially involves placing *implications* like $\beta \rightarrow \alpha$ higher in the ranking than α. Using the ranking in this way helps to overcome certain problems with material implication such as those that arise when the antecedent is *false*, for instance. However, the identification of reasons in this simple-minded way presents certain difficulties, such as enumerating all the exceptions in the antecedent (McCarthy, 1980). Without wishing to discount this problem we simply say that for the purpose of modeling change, *not* being able to explicitly represent reasons means that more information will be retained during a maxi-adjustment since information is retracted only if there *is* a reason to remove it.

When contracting α a maxi-adjustment retracts β if $\text{degree}(\mathbf{B}, \alpha) < \text{degree}(\mathbf{B}, \beta \rightarrow \alpha)$. The standard construction of Gärdenfors and Makinson uses $\text{degree}(\mathbf{B}, \alpha) = \text{degree}(\mathbf{B}, \alpha \vee \beta)$. It turns out that $\text{degree}(\mathbf{B}, \alpha) < \text{degree}(\mathbf{B}, \beta \rightarrow \alpha)$ implies $\text{degree}(\mathbf{B}, \alpha) = \text{degree}(\mathbf{B}, \alpha \vee \beta)$ in which case both adjustment and maxi-adjustment retract β.

In the case that $\text{degree}(\mathbf{B}, \alpha) = \text{degree}(\mathbf{B}, \beta \rightarrow \alpha) = \text{degree}(\mathbf{B}, \alpha \vee \beta)$, a maxi-adjustment retracts all β's in $\text{Cn}(\alpha)$ not entailed by other beliefs. This results in the retraction of $\beta \rightarrow \alpha$ and the retention of β. Ordinary adjustment would retract both $\beta \rightarrow \alpha$ and β, hence maxi-adjustments conserve more information. Our justification for this process is: (i) if $\text{degree}(\mathbf{B}, \alpha) = \text{degree}(\mathbf{B}, \alpha \vee \beta)$ then β is *not* a reason for α, and hence should be retained if possible, and (ii) $\text{degree}(\mathbf{B}, \alpha) = \text{degree}(\mathbf{B}, \beta \rightarrow \alpha)$ suggests that $\beta \rightarrow \alpha$ does not provide more *epistemic power* than α itself. If this behaviour is not desirable then the definition of maxi-adjustment can be trivally modified so that beliefs ranked at $\text{degree}(\mathbf{B}, \alpha)$ are treated in the same fashion as beliefs ranked strictly less than $\text{degree}(\mathbf{B}, \alpha)$. This results in similar procedure to Nebel's priority base construction, and the change operators in Theorem 2 satisfy all the AGM postulates for revision and withdrawal without requiring the condition in Theorem 3.

Finally, we retain sentences β when $\mathbf{B}(\beta) < \text{degree}(\mathbf{B}, \alpha)$ so long as it does not give rise to a reason for α.

The main advantage of maxi-adjustments over adjustments is that beliefs are retained without the need to include a host of disjunctions in the domain of the ranking. Without the inclusion of such disjunctions the adjustment procedure will ruthlessly discard beliefs.

Alchourrón *et al*'s (1985) partial meet contraction functions T^-_α are constructed from maximal subsets of T that do not entail α, whilst Alchourrón and Makinson's (1985) safe contraction functions are constructed using an acyclical hierarchy over T and the inherited hierarchy over minimal subsets that entail α. Since the maxi-adjustment procedure computes minimal subsets that entail α at each rank of $\mathbf{B}$, more work needs to be done to identify the exact nature of the connection between safe contraction and maxi-adjustment.

An important property of both adjustments and maxi-adjustments is that they can be used to obtain any finite partial ranking. For a finite language every ranking is *reachable* via a finite number of maxi-adjustments, or a finite number of adjustments.

Just like adjustments, maxi-adjustments use the *qualitative information* encoded in a partial entrenchment ranking, i.e. the relative ordering of sentences, and they *preserve finiteness*; adjusting a finite partial entrenchment ranking results in a finite ranking, and if $\text{content}(\mathbf{B})$ is finite then $\text{content}(\mathbf{B}^\star(\alpha, i))$ is

finite. Note that both an adjustment and a maxi-adjustment, $\mathbf{B}^*(\alpha, i)$, leave all explicit information assigned ordinals greater than $\max(\{i, \mathbf{B}(\alpha), \mathbf{B}(\neg\alpha)\})$ $\mathbf{B}$ unchanged.

In an attempt to mesh Gärdenfors interpretation of Spohnian reasons based on dependence and our notion of Spohnian reasons based on maxi-adjustments we can say: For α and β in content($\mathbf{B}$) if β is a reason α then β is dependent on α using the standard contraction construction based on epistemic entrenchments.

5 AN APPLICATION

Maxi-adjustments have been successfully used to model changes to user requirements in an *Aircraft Landing System* application (MacNish and Williams, 1996). In this section we demonstrate that maxi-adjustments are suitable for modeling changes to rules and facts in prioritised knowledge bases using a simple *Financial Advisory System* (Luger and Stubblefield, 1993). The system takes information about an individual's income and savings, and then proposes an appropriate investment plan.

In most cases using an adjustment to achieve the same result as maxi-adjustment would require the inclusion of a large number of pairwise disjunctions specifying the independence of explicit information.

For the application described below this would mean the addition of 45 extra sentences to the domain of the ranking. Note that all the changes described in this example satisfy *all* the AGM postulates for revision and withdrawal.

For convenience, and without loss of generality where the domain of the ranking is finite, we map sentences to rational numbers in [0, 1] instead of ordinals.

$\mathbf{B}(savings(inadequate) \rightarrow investment(savings\_acct)) = 0.8,$

$\mathbf{B}(savings(adequate) \wedge income(adequate) \rightarrow investment(stocks)) = 0.8,$

$\mathbf{B}(savings(adequate) \wedge income(inadequate) \rightarrow investment(combination)) = 0.8,$

$\mathbf{B}(\forall X\ amount\_saved(X) \wedge \exists Y(dependents(Y) \wedge greater(X, min\_savings(Y))) \rightarrow savings(adequate)) = 0.8,$

$\mathbf{B}(\forall X\ amount\_saved(X) \wedge \exists Y(dependents(Y) \wedge \neg greater(X, min\_savings(Y))) \rightarrow savings(inadequate)) = 0.8,$

$\mathbf{B}(\forall X\ earnings(X, steady) \wedge \exists Y(dependents(Y) \wedge greater(X, min\_income(Y))) \rightarrow income(adequate)) = 0.8,$

$\mathbf{B}(\forall X\ earnings(X, steady) \wedge \exists Y(dependents(Y) \wedge \neg greater(X, min\_income(Y))) \rightarrow income(inadequate)) = 0.8,$

$\mathbf{B}(\forall X\ earnings(X, unsteady) \rightarrow savings(inadequate)) = 0.8,$

$\mathbf{B}(amount\_saved(22,000)) = 0.5,$

$\mathbf{B}(earnings(25,000, steady)) = 0.5,$

$\mathbf{B}(dependents(4)) = 0.3,$

where $min\_savings(X) = 5,000 * X$ and where $min\_income(X) = 15,000 + (4,000 * X)$. Let us assume that we have also encoded integrity constraints with degree 0.9 that say there can only be one occurence of the predicates: *amount_saved*, *earnings*, *dependents*.

Using the function degree we can calculate:
degree($\mathbf{B}$, *investment*(*combination*)) = 0.3,
degree($\mathbf{B}$, *investment*(*stocks*)) = 0, and
degree($\mathbf{B}$, *investment*(*saving_acct*)) = 0.

(a) Let us now revise the knowledge base. Consider $\mathbf{B}^*(earnings(50,000), 0.5)$. If a maxi-adjustment is used then the sentence $earnings(25,000, steady)$ will be replaced with $earnings(50,000, steady)$ with the degree of acceptance 0.5, and all other assignments will be preserved. As a result we have
degree($\mathbf{B}$, *investment*(*stocks*)) = 0.3,
degree($\mathbf{B}$, *investment*(*combination*)) = 0, and
degree($\mathbf{B}$, *investment*(*saving_acct*)) = 0.

In contrast if an adjustment is used then not only will $earnings(50,000, steady)$ be assigned 0.5, and
$\mathbf{B}^*(earnings(50,000, steady), 0.5)$
$(amount\_saved(25,000, steady)) = 0$ and
$\mathbf{B}^*(earnings(50,000, steady), 0.5)(dependents(4)) = 0.$
This means that a change in an individual's earnings results in the retraction of their savings and the number of their dependents. In other words, changing an individual's salary causes a substantial loss information.

(b) Removing one of the rules assigned 0.8 using a maxi-adjustment will not effect the rank of any other rule or fact. However if adjustment is used then all rules and facts will be assigned zero because their independence has not been specified in the ranking.

(c) Removing *investment*(*combination*) using a maxi-adjustment results in *dependents*(4) being reassigned zero, and all other sentences remaining as they were previously ranked, i.e. only the weakest conjunct in the reason is removed.
Note, an adjustment in this case would give the same result. The reason for this is that an adjustment will preserve the ranking above 0.3. If there were sentences with a rank less than or equal to 0.3 then they would be vunerable.

(d) If *investment*(*combination*) was an explicit belief ranked at 0.5 then a maxi-adjustment remove *investment*(*combination*) will behave as in (c). However an adjustment would retract all facts ranked below 0.5, i.e. adjustment would retract *amount_saved*(22,000), *earnings*(25,000, *steady*), and *dependents*(4).

6 DISCUSSION

We introduced a maxi-adjustment which is a new method for modifying a partial entrenchment ranking. It retains as much accepted information as possible under the assumption that if β is a reason for α then β should be removed in the contraction of α, and if $\gamma \rightarrow \alpha$ possesses the same rank as α then it should be removed in preference to γ because according to our interpretation of Spohnian reasons γ is not a reason for α and $\gamma \rightarrow \alpha$ does not provide more epistemic power than α itself.

We have shown that Spohnian reasons can not only be used for problem solving but also for specifying change. Maxi-adjustments were motivated by the need to develop belief revision techniques that provide for applications in which it is less demanding for the system designer to specify dependence than independence.

Maxi-adjustments have been used successfully in modeling changes to user requirements in a Software Engineering context, and they are currently being implemented in C^{++} as part of the Intelligent Information Management Toolkit under development in the CIN Project (Antoniou and Williams, 1996).

The idea behind the Belief Revision module of the Toolkit is to offer a systems designer several methods for specifying and revising a ranking. For example, if the system designer is in a position to specify the independencies among beliefs then an adjustment will modify the original ranking in an absolute minimal way. However, if specifying dependencies is less demanding, then the designer can use a maxi-adjustment which modifies the ranking in an absolute minimal way under maximal inertia.

Given the transparent connection between belief revision and nonmonotonic reasoning established by Gärdenfors and Makinson (1994) maxi-adjustments can also be used to modify rational nonmonotonic inference relations along the lines of the approach taken by Williams (1995b) in a more natural way than adjustments.

Acknowledgements

The author gratefully acknowledges Hans Rott and Alvaro del Val for their animadvert discussions on the philosophical practicalities of belief revision. Thanks are also due to Grigorios Antoniou, Chitta Baral, Norman Foo, Kym MacNish, Leora Morgenstern, Bernhard Nebel and Pavlos Peppas. The research reported in this paper was funded by the Australian Research Council.

References

Alchourrón, C., and Makinson, D. (1985), *On the logic of theory change: Safe Contractions*, Studia Logica 44: 405 - 422, 1985.

Alchourrón, C., Gärdenfors, P., and Makinson, D. (1985), *On the logic of theory change: Partial meet functions for contraction and revision*, Journal of Symbolic Logic 50: 510 - 530, 1985.

Antoniou, G. and Williams, M.A. (1996), *Defaut Reasoning and Belief Revision in the CIN Project*, in the Proceedings of the First International Conference on Formal and Practical Reasoning, LNCS, Springer Verlag, 691 – 694, 1996.

Boutilier, C. (1993), *Revision Sequences and Nested Conditionals*, in Proceedings IJCAI, Morgan Kaufmann, 519 – 525, 1993.

Darwiche, D. and Pearl, J. (1994), *On the Logic of Iterated Belief Revision*, in Fagin (ed), in Proceedings of TARK, 5 – 23, 1994.

Dubois, D. and Prade, H. (1994), *Possibilistic Logic*, Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 3, Nonmonotonic Reasoning and Uncertain Reasoning, Gabbay, D., Hogger, C, and Robinson, J. (eds), Claredon Press, Oxford, 1994.

Freund, M. and Lehmann, D. (1994), *Belief Revision and Rational Inference*, Technical Report TR 94–16, Institute of Computer Science, The Hebrew University of Jerusalem, Israel, 1994.

Fuhrmann, A. (1991), *Theory Contraction through Base Contraction*, Journal of Philosophical Logic, 20: 175 – 203, 1991.

Gärdenfors, P. (1988), *Knowledge in Flux*, A Bradford Book, The MIT Press, 1988.

Gärdenfors, P. (1992), *The Dynamics of Belief Systems: Foundations versus Coherence Theories*, in Knowledge, Belief, and Strategic Interaction, Cristina Bicchieri, Maria Luisa Dalla Chiara (eds) Cambridge University Press, 1992.

Gärdenfors, P., and Makinson, D. (1988), *Revisions of Knowledge Systems using Epistemic Entrenchment*, Proceedings of TARK, 83 - 96, 1988.

Gärdenfors, P. and Makinson, D. (1994), *Nonmonotonic Inference Based on Expectations*, Artificial Intelligence Journal 65: 197 - 245, 1994.

Hansson, S.O. (1989), *New Operators for Theory Change*, Theoria 55: 115 - 132, 1989.

Hansson, S.O. (1993), *Reversing the Levi Identity*, journal of Philosophical Logic, 22: 637 - 669, 1993.

Lehmann, D. (1995), *Belief Revision, Revised*, in the Proceedings of Fourteenth IJCAI, Morgan Kaufmann, 1995.

Luger, G. and Stubblefield, W.A. (1993), *Artifical Intelligence: Structures and Strategies for Complex Problem Solving*, Benjamin Cummings, 1993.

McCarthy, J. (1980), *Circumscription - a form of Nonmonotonic Reasoning*, Journal of Artificial Intelligence 13: 27 - 39, 1980.

MacNish, C.K. and Williams M.A. (1996), *From Belief Revision to Design Revision: Applying Theory Change to Changing Requirements*, Pacific Rim Workshop on Incomplete and Changing Information, 1996 (in press).

Makinson, D. (1987), *On the Status of the Postulate of Recovery in the Logic of Theory Change*, Journal of Philosophical Logic, 16: 383 - 394, 1987.

Nayak, A. (1994), *Iterated Belief Change based on Epistemic Entrenchment*, Erkenntnis, 14: 353 - 390, 1994.

Nebel, B., (1989) *A Knowledge Level Analysis of Belief Revision*, in Principles of Knowledge Representation and Reasoning: Proceedings of the International Conference, Morgan Kaufmann, San Mateo, CA, 301 - 311, 1989.

Nebel, B., (1991) *Belief Revision and Default Reasoning*, in Principles of Knowledge Representation and Reasoning: Proceedings of the International Conference, Morgan Kaufmann, San Mateo, CA, 417 - 428, 1991.

Pearl, J. (1994), *From Adams' Conditionals to Default Expressions, Causal Conditionals, and Counterfactuals*, in E. Eells and B. Skyrms (eds), *Probability and Conditionals*, Cambridge University Press, 47 - 74, 1994.

H. Rott, (1991) *A Nonmonotonic Conditional Logic for Belief Revision I*, in A. Fuhrmann and M.Morreau (eds), The Logic of Theory Change, Springer-Verlag, LNAI 465, Berlin, 135 - 183, 1991.

Spohn, W., (1983) *Deterministic and Probabilistic Reasons*, Erkenntis 19: 371 - 396, 1983.

Spohn, W., (1988) *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.

M.A. Williams, (1994a) *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.

M.A. Williams,(1994b) *On the Logic of Theory Base Change*, in Logics in Artificial Intelligence, C. MacNish, D. Pearce and L.M. Perira (eds), LNCS No 835, 86 - 105, Springer Verlag, 1994.

M.A. Williams, (1995a) *Iterated Theory Base Change: A Computational Model*, in the Proceedings of the Fourteenth IJCAI, Morgan Kaufmann, 1541 - 1550, 1995.

M.A. Williams, (1995b) *Changing Nonmontonic Reasoning Inference Relations*, in the Proceedings of the Second World Conference on the Fundamentals of Artifical Intelligence, 469 - 482, Ankgor, Paris, 1995.

M.A. Williams, M. Pagnucco, N.Y. Foo, B. Sims, (1995a) *Determining Explanations using Transmutations*, in the Proceedings of the Fourteenth IJCAI, Morgan Kaufmann, 822 - 830,1995.

M.A. Williams, K. Wallace, G. Antoniou, (1995b) *An Object-Oriented Implementation of Belief Revision*, in the Proceedings of the Australian Joint Conference on Artificial Intelligence, World Scientific, 259 - 267, 1995.

Belief Revision: A Critique

Nir Friedman*
Computer Science Department
Stanford University
Gates Building 1A
Stanford, CA 94305-9010
nir@cs.stanford.edu

Joseph Y. Halpern†
IBM Research Division
Almaden Research Center, Dept. K53-B2
650 Harry Road
San Jose, CA 95120-6099
halpern@almaden.ibm.com

Abstract

We examine carefully the rationale underlying the approaches to belief change taken in the literature, and highlight what we view as methodological problems. We argue that to study belief change carefully, we must be quite explicit about the "ontology" or scenario underlying the belief change process. This is something that has been missing in previous work, with its focus on postulates. Our analysis shows that we must pay particular attention to two issues that have often been taken for granted: The first is how we model the agent's epistemic state. (Do we use a set of beliefs, or a richer structure, such as an ordering on worlds? And if we use a set of beliefs, in what language are these beliefs are expressed?) We show that even postulates that have been called "beyond controversy" are unreasonable when the agent's beliefs include beliefs about her own epistemic state as well as the external world. The second is the status of observations. (Are observations known to be true, or just believed? In the latter case, how firm is the belief?) Issues regarding the status of observations arise particularly when we consider *iterated* belief revision, and we must confront the possibility of revising by φ and then by $\neg\varphi$.

*Current address: Computer Science Division, University of California, Berkeley CA 94720.

†Current address: Computer Science Department, Cornell University, Ithaca NY 14853.

1 INTRODUCTION

The problem of *belief change*—how an agent should revise her beliefs upon learning new information—has been an active area of research in both philosophy and artificial intelligence. The problem is a fascinating one in part because it is clearly no unique answer. Nevertheless, there is a strong intuition that one wants to make *minimal* changes, and all the approaches to belief change in the literature, such as [Alchourrón, Gärdenfors, and Makinson 1985; Gärdenfors 1988; Katsuno and Mendelzon 1991a], try to incorporate this principle. However, approaches differ on what constitutes a minimal change. This issue has come to the fore with the spate of recent work on *iterated* belief revision (see, for example, [Boutilier 1993; Boutilier and Goldszmidt 1993; Darwiche and Pearl 1994; Freund and Lehmann 1994; Lehmann 1995; Levi 1988; Williams 1994]).

The approaches to belief change typically start with a collection of postulates, argue that they are reasonable, and prove some consequences of these postulates. Occasionally, a semantic model for the postulates is provided and a representation theorem is proved (of the form that every semantic model corresponds to some belief revision process, and that every belief revision process can be captured by some semantic model). Our goal in this paper is not to introduce yet another model of belief change, but to examine carefully the rationale underlying the approaches in the literature. The main message of the paper is that describing postulates and proving a representation theorem is not enough. While it may have been reasonable when research on belief change started in the early 1980s to just consider the implications of a number of seemingly reasonable postulates, it is our view that it should no longer be acceptable just to write down postulates and give short English justi-

fications for them. While postulates do provide insight and guidance, it is also important to describe what, for want of a better word, we call the underlying *ontology* or scenario for the belief change process. Roughly speaking, this means describing carefully what it means for something to be believed by an agent and what the status is of new information that is received by the agent. This point will hopefully become clearer as we present our critique. We remark that even though the issue of ontology is tacitly acknowledged in a number of papers (for example, in the last paragraph of [Lehmann 1995]), it rarely enters into the discussion in a significant way. We hope to show that ontology must play a central role in all discussions of belief revision.

Our focus is on approaches that take as their starting point the postulates for belief revision proposed by Alchourrón, Gärdenfors, and Makinson (AGM from now on) [1985], but our critique certainly applies to other approaches as well, in particular, Katsuno and Mendelzon's *belief update* [1991b]; see Section 5. The AGM approach assumes that an agent's epistemic state is represented by a *belief set*, that is, a set K of formulas in a logical language $\mathcal{L}$. What the agent learns is assumed to be characterized by some formula φ, also in $\mathcal{L}$; $K * \varphi$ describes the belief set of an agent that starts with belief set K and learns φ.

There are two assumptions implicit in this notation:

- The functional form of $*$ suggests that all that matters regarding how an agent revises her beliefs is the belief set and what is learnt.
- The notation suggests that the second argument of $*$ can be an arbitrary formula in $\mathcal{L}$. But what does it mean to revise by *false*? In what sense can *false* be learnt? More generally, is it reasonable to assume that an arbitrary formula can be learnt in a given epistemic state?

The first assumption is particularly problematic when we consider the postulates that AGM require $*$ to satisfy. These essentially state that the agent is consistent in her choices, in the sense that she acts as though she has an ordering on the strength of her beliefs [Gärdenfors and Makinson 1988; Grove 1988], or an ordering on possible worlds [Boutilier 1994; Grove 1988; Katsuno and Mendelzon 1991b], or some other predetermined manner of choosing among competing beliefs [Alchourrón, Gärdenfors, and Makinson 1985]. However, the fact that an agent's epistemic state is characterized by a collection of formulas means that the epistemic state cannot include information about relative strength of beliefs (as required for the approach of, say, [Gärdenfors and Makinson 1988]), unless this information is expressible in the language. Note that if $\mathcal{L}$ is propositional logic or first-order logic, such information cannot be expressed. On the other hand, if $\mathcal{L}$ contains *conditional* formulas of the form $p > q$, interpreted as "if p is learnt, then q will be believed", then constraints on the relative strength of beliefs can be expressed (indirectly, by describing which beliefs will be retained after a revision).

Problems arise when the language is not rich enough to express relative degrees of strength in beliefs. Consider, for example, a situation where $K = Cl(p \wedge q)$ (the logical closure of $p \wedge q$; that is, the agent's beliefs are characterized by the formula $p \wedge q$ and its logical consequences), and then the agent learns $\varphi = \neg p \vee \neg q$. We can imagine that an agent whose belief in p is stronger than her belief in q would have $K * \varphi = \{p\}$. That is, the agent gives up her belief in q, but retains a belief in p. On the other hand, if the agent's belief in q is stronger than her belief in p, it seems reasonable to expect that $K * \varphi = \{q\}$. This suggests that it is unreasonable to take $*$ to be a function if the representation language is not rich enough to express what may be significant details of an agent's epistemic state.

We could, of course, assume that information about the relative strength of beliefs in various propositions is implicit in the choice of the revision operator $*$, even if it is not contained in the language. This is perfectly reasonable, and also makes it more reasonable that $*$ be a function. However, note that we can then no longer assume that we use the same $*$ when doing iterated revision, since there is no reason to believe that the relative strength of beliefs is maintained after we learn a formula. In fact, in a number of recent papers [Boutilier 1993; Boutilier and Goldszmidt 1993; Friedman and Halpern 1995a; Williams 1994], $*$ is defined as a function from (epistemic states $\times$ formulas) to epistemic states, but the epistemic states are no longer just belief sets; they include information regarding relative strengths of beliefs. The revision function on epistemic states induces a mapping from (belief sets $\times$ formulas) to belief sets, but at the level of belief sets, the mapping may not be functional; for a belief set K and formula φ, the belief set $K * \varphi$ may depend on what epistemic state induced K. Thus, the effect of $*$ on belief sets may change over time.[1]

[1] Freund and Lehmann [1994] have called the view-

There is certainly no agreement on what postulates belief change should satisfy. However, the following two postulates are almost universal:

- $\varphi \in K * \varphi$
- if K is consistent and $\varphi \in K$, then $K * \varphi = K$.

These postulates have been characterized by Rott [1989] as being "beyond controversy". Nevertheless, we argue that they are not as innocent as they may at first appear.

The first postulate says that the agent believes the last thing she learns. Making sense of this requires some discussion of the underlying ontology. It certainly makes sense if we take (as Gärdenfors [1988] does) the belief set K to consist of all formulas that are accepted by the agent (where "accepted" means "treated as true"), and the agent revises by φ only if φ has somehow come to be accepted. However, note that deciding when a formula has come to be accepted is nontrivial. In particular, just observing φ will not in general be enough for accepting φ. Acceptance has a complex interaction with what is already believed. For example, imagine a scientist who believes that heavy objects drop faster than light ones, climbs the tower of Pisa, drops a 5 kilogram textbook and a 500 milligram novel, and observes they hit the ground at the same time. This scientist will probably not accept that the time for an object to fall to the ground is independent of its weight, on the basis of this one experiment (although perhaps repeated experiments may lead her to accept it).

While the acceptance point of view is certainly not unreasonable, the fact that just observing φ is not necessarily enough for acceptance often seems forgotten. It also seems hard to believe that *false* would ever be accepted. More generally, it is far from obvious that in a given epistemic state K we should allow arbitrary consistent formulas to be accepted. Intuitively, this does not allow for the possibility that some beliefs are held so firmly that their negations could never be accepted. (Later we describe an ontology where observations are taken to be known in which in fact some consistent formulas will not be accepted in some epistemic states.)

While the second postulate is perhaps plausible if we cannot talk about epistemic importance or strength of belief in the language, it is less so once we can talk about such things (or if either epistemic importance or strength of belief is encoded in the epistemic state some other way). For suppose that $\varphi \in K$. Why should $K * \varphi = K$? It could well be that being informed of φ raises the importance of φ in the epistemic ordering, or the agent's strength of belief in φ. If strength of belief can be talked about in the language, then a notion of minimal change should still allow strengths of belief to change, even when something expected is observed. Even if we cannot talk about strength of belief in the language, this observation has an impact on iterated revisions. For example, one assumption made by Lehmann [1995] (his postulate I4) is that if p is believed after revising by φ, then revising by $[\varphi \cdot p \cdot \psi]$—that is, revising by φ then p then ψ—is equivalent to revising by $[\varphi \cdot \psi]$. But consider a situation where after revising by φ, the agent believes both p and q, but her belief in q is stronger than her belief in p. We can well imagine that after learning $\neg p \vee \neg q$ in this situation, she would believe $\neg p$ and q. However, if she first learned p and then $\neg p \vee \neg q$, she would believe p and $\neg q$, because, as a result of learning p, she would give p higher epistemic importance than q. In this case, we would not have $[\varphi \cdot p \cdot (\neg p \vee \neg q)] = [\varphi \cdot (\neg p \vee \neg q)]$. In light of this discussion, it is not surprising that the combination of the second postulate with a language that can talk about epistemic ordering leads to technical problems such as Gärdenfors' *triviality result* [1988].

To give a sense of our concerns here, we discuss two basic ontologies. The first ontology that seems (to us) reasonable assumes that the agent has some knowledge as well as beliefs. We can think of the formulas that the agent knows as having the highest state of epistemic importance. In keeping with the standard interpretation of knowledge, we also assume that the formulas that the agent knows are true in the world. Since agents typically do not have certain knowledge of very many facts, we assume that the knowledge is augmented by beliefs (which can be thought of as defeasible guides to action). Thus, the set of formulas that are known form a subset of the belief set. We assume that the agent observes the world using reliable sensors; thus, if the agent observes φ, then the agent is assumed to know φ. After observing φ, the agent adds φ to his stock of knowledge, and may revise his belief set. Since the agent's observations are taken to be knowledge, the agent will believe φ after observing φ. However, the agent's epistemic state may change even if she observes a formula

point that $*$ may change over time the *dynamic* point of view. However, this seems somewhat of a misnomer when applied to papers such as [Boutilier 1993; Boutilier and Goldszmidt 1993; Friedman and Halpern 1995a; Williams 1994], since there $*$ in fact is static, when viewed as a function on epistemic states and formulas.

that she previously believed to be true. In particular, if the formula observed was believed to be true but not known to be true, after the observation it is known. Note that, in this ontology, the agent never observes *false*, since *false* is not true of the world. In fact, the agent never observes anything that contradicts her knowledge. Thus, $K * \varphi$ is defined only for formulas φ that are compatible with the agent's knowledge. Moving to iterated revision, this means we cannot have a revision by φ followed by a revision by $\neg\varphi$. This ontology underlies some of our earlier work [1995b, 1995a]. As we show here, a variant of Darwiche and Pearl's approach [1994] captures them as well.

We can consider a second ontology that has a different flavor. In this ontology, if we observe something, we believe it to be true and perhaps even assign it a strength of belief. But this assignment does not represent the strength of belief of the observation in the resulting epistemic state. Rather, the belief in the observation must "compete" against current beliefs if it is inconsistent with these beliefs. In this ontology, it is not necessarily the case that $\varphi \in K * \varphi$, just as it is not the case that a scientist will necessarily adopt the consequences of his most recent observation into his stock of beliefs (at least, not without doing some additional experimentation). Of course, to flesh out this ontology, we need to describe how to combine a given strength of belief in the observation with the strengths of the beliefs in the original epistemic state. Perhaps the closest parallel in the literature is something like the Dempster-Shafer rule of combination [Shafer 1976], which gives a rule for combining two separate bodies of belief. We do not have a particular suggestion to make along these lines. However, we believe that this type of ontology deserves further study.

The rest of the paper is organized as follows. In Section 2, we review the AGM framework, and point out some problems with it. In Section 3, we consider proposals for belief change and iterated belief change from the literature due to Boutilier [1993], Darwiche and Pearl [1994], Freund and Lehmann [1994], and Lehmann [1995], and try to understand the ontology implicit in the proposal (to the extent that one can be discerned). In Section 4, we consider the first ontology discussed above in more detail. We conclude with some discussion in Section 5.

2 AGM BELIEF REVISION

In this section we review the AGM approach to belief revision. As we said earlier, this approach assumes that beliefs and observations are expressed in some language $\mathcal{L}$. It is assumed that $\mathcal{L}$ is closed under negation and conjunction, and comes equipped with a consequence relation $\vdash_{\mathcal{L}}$ that contains the propositional calculus and satisfies the deduction theorem. The agent's epistemic state is represented by a belief set, that is, a set of formulas in $\mathcal{L}$ closed under deduction. There is also assumed to be a revision operator $*$ that takes a belief set K and a formula φ and returns a new belief set $K * \varphi$, intuitively, the result of revising K by φ. The following AGM postulates are an attempt to characterize the intuition of "minimal change":

R1. $K * \varphi$ is a belief set
R2. $\varphi \in K * \varphi$
R3. $K * \varphi \subseteq Cl(K \cup \{\varphi\})$
R4. If $\neg\varphi \notin K$ then $Cl(K \cup \{\varphi\}) \subseteq K * \varphi$
R5. $K * \varphi = Cl(\mathit{false})$ if and only if $\vdash_{\mathcal{L}} \neg\varphi$
R6. If $\vdash_{\mathcal{L}} \varphi \Leftrightarrow \psi$ then $K * \varphi = K * \psi$
R7. $K * (\varphi \wedge \psi) \subseteq Cl(K * \varphi \cup \{\psi\})$
R8. If $\neg\psi \notin K * \varphi$ then $Cl(K * \varphi \cup \{\psi\}) \subseteq K * (\varphi \wedge \psi)$

The essence of these postulates is the following. Revising K by φ gives a belief set (Postulate R1) that includes φ (R2). If φ is consistent with K, then $K * \varphi$ should not remove any of the old beliefs and should not add any new beliefs except these implied by the combination of the old beliefs with the new belief (R3 and R4). Note that it follows from R3 that if $\varphi \in K$, then $\varphi \in K * \varphi$. The next two postulates discuss the coherence of beliefs. R5 states that as long as φ is consistent, then so is $K * \varphi$, and R6 states that the syntactic form of the new belief does not affect the revision process. The last two postulates enforce a certain coherency on the outcome of revisions by related beliefs. Basically they state that if ψ is consistent with $K * \varphi$ then $K * (\varphi \wedge \psi)$ is the result of adding ψ to $K * \varphi$.

The intuitions described by AGM is based on one-step (noniterated) revision. Nevertheless, the AGM postulates do impose some restrictions on iterated revisions. For example, suppose that q is consistent with $K * p$. Then, according to R2 and R3, $(K * p) * q = \mathrm{Cl}(K * p \cup \{q\})$. Using R7 and R8 we can conclude that $(K * p) * q = K * (p \wedge q)$.

There are several representation theorems for AGM belief revision; perhaps the clearest is due to Grove [1988]. We discuss a slight modification, due to Boutilier [1994] and Katsuno and Mendelzon [1991b]: Let an *$\mathcal{L}$-world* be a complete and consistent truth assignment to the formulas in $\mathcal{L}$. Let $\mathcal{W}$ consist of all the $\mathcal{L}$-worlds, and let $\preceq$ be

a *ranking*, that is, a total preorder, on the worlds in $\mathcal{W}$. Let $\min_{\preceq}$ consist of all the minimal worlds with respect to $\preceq$, that is, all the worlds w such that there is no w' with $w' \prec w$. With $\preceq$ we can associate a belief set $\mathrm{Bel}(\preceq)$, consisting of all formulas φ that are true in all the worlds in $\min_{\preceq}$. Moreover, we can define a revision operator $*$ on $\mathrm{Bel}(\preceq)$, by taking $\mathrm{Bel}(\preceq) * \varphi$ to consist of all formulas ψ that are true in all the minimal φ-worlds according to $\preceq$. It can be shown that $*$ satisfies the AGM postulates (when its first argument is $\mathrm{Bel}(\preceq)$). Thus, we can define a revision operator by taking a collection of orderings $\preceq_K$, one for each belief set K. To define $K * \varphi$ for a belief set K, we apply the procedure above, starting with the ranking $\preceq_K$ corresponding to K.[2] Furthermore, Grove [1988], Katsuno and Mendelzon [1991b], and Boutilier [1994] show that every belief revision operator satisfying the AGM axioms can be characterized in this way.

This elegant representation theorem also brings out some of the problems with the AGM postulates. First, note that a given revision operator $*$ is represented by a family of rankings, one for each belief set. There is no necessary connection between the rankings corresponding to different belief sets. It might seem more reasonable to have a more global setting (perhaps one global ranking) from which each element in the family of rankings arises.

A second important point is that the epistemic state here is represented not by a belief set, but by a ranking. Each ranking $\preceq$ is associated with a belief set $\mathrm{Bel}(\preceq)$, but it is the ranking that gives the information required to describe how revision is carried out. The belief set does not suffice to determine the revision; there are many rankings $\preceq$ for which the associated belief set $\mathrm{Bel}(\preceq)$ is K. Since the revision process only gives us the revised belief set, not the revised ranking, the representation does not support iterated revision.

This suggests that we should consider, not how to revise belief sets, but how to revise rankings. More generally, whatever we take to be our representation of the epistemic state, it seems appropriate to consider how these representations should be revised. We can define an analogue of the AGM postulates for epistemic states in a straightforward way (cf. [Friedman and Halpern 1995a]): Taking E to range over epistemic states and $Bel(E)$ to represent the belief set associated with epistemic state E, we have

R1′. $E * \varphi$ is an epistemic state
R2′. $\varphi \in Bel(E * \varphi)$
R3′. $Bel(E * \varphi) \subseteq Cl(Bel(E) \cup \{\varphi\})$

and so on, with the obvious syntactic transformation. In fact, as we shall see in the next section, a number of processes for revising epistemic states have been considered in the literature, and in fact they all do satisfy these modified postulates.

Finally, even if we restrict attention to belief sets, we can consider what happens if the underlying language $\mathcal{L}$ is rich enough to talk about how revision should be carried out. For example, suppose $\mathcal{L}$ includes conditional formulas, and we want to find some ranking $\preceq$ for which the corresponding belief set is K. Not just any ranking $\preceq$ such that $\mathrm{Bel}(\preceq) = K$ will do here. The beliefs in K put some constraints on the ranking. For example, if $p > q$ is in K and $p \notin K$, then the minimal $\preceq$-worlds satisfying p must all satisfy q, since after p is learnt, q is believed. Once we restrict to rankings that are consistent with the formulas in the worlds that are being ranked, then the AGM postulates are no longer sound. This point has essentially been made before [Boutilier 1992; Rott 1989]. However, it is worth stressing the sensitivity of the AGM postulates to the underlying language and, more generally, to the choice of epistemic state.

3 PROPOSALS FOR ITERATED REVISION

We now briefly review some of the previous proposals for iterated belief change, and point out how the impact of the observations we have been making on the approaches. Most of these approaches start with the AGM postulates, and augment them to get seemingly appropriate restrictions on iterated revision. This is not an exhaustive review of the literature on iterated belief revision by any stretch of the imagination. Rather, we have chosen a few representative approaches that allow us to bring out our methodological concerns.

3.1 BOUTILIER'S NATURAL REVISION

As we said in the previous section, Boutilier takes the agent's epistemic state to consist of a ranking of possible worlds. Boutilier [1993] describes a particular revision operator $*_B$ on epistemic states that he calls *natural revision* operator. Natural revision maps a ranking $\preceq$ of possible worlds and

[2] In this construction, for each belief set K other than the inconsistent belief set, we have $\mathrm{Bel}(\preceq_K) = K$. The inconsistent belief set gets special treatment here.

an observation φ to a revised ranking $\preceq *_B \varphi$ such that (a) $\preceq *_B \varphi$ satisfies the conditions of the representation theorem described above—the minimal worlds in $\preceq *_B \varphi$ are precisely the minimal φ-worlds in $\preceq$, and (b) in a precise sense, $\preceq *_B \varphi$ is the result of making the minimal number of changes to $\preceq$ required to guarantee that all the minimal worlds in $\preceq *_B \varphi$ satisfy φ. Given a ranking $\preceq$ and a formula φ, the ranking $\preceq *_B \varphi$ is identical to $\preceq$ except that the minimal φ-worlds according to $\preceq$ have the minimal rank in the revised ranking, while the relative ranks of all other worlds remains unchanged.

Boutilier characterizes the properties of natural revision. Suppose that, starting in some epistemic state, we revise by $\varphi_1, \ldots, \varphi_n$. Further suppose φ_{i+1} is consistent with the beliefs after revising by $\varphi_1, \ldots, \varphi_i$. Then the beliefs after revising by $\varphi_1, \ldots, \varphi_n$ are precisely the beliefs after observing $\varphi_1 \wedge \ldots \wedge \varphi_n$. (More precisely, given any ranking $\preceq$, the belief set associated with the ranking $\preceq *_B \varphi_1 *_B \ldots *_B \varphi_n$ is the same as that associated with the ranking $\preceq *_B (\varphi_1 \wedge \ldots \wedge \varphi_n)$. Note, however, that $\preceq *_B \varphi_1 *_B \ldots *_B \varphi_n \neq \preceq *_B (\varphi \wedge \ldots \wedge \varphi_n)$ in general.) Thus, as long as the agent's new observations are not surprising, the agent's beliefs are exactly the ones she would have had had she observed the conjunction of all the observations. This is an immediate consequence of the AGM postulates, and thus holds for any approach that attempts to extend the AGM postulates to iterated revision.

What happens when the agent observes a formula φ_{n+1} that is inconsistent with her current beliefs? Boutilier shows that in this case the new observation nullifies the impact of the all the observations starting with the most recent one that is inconsistent with φ_{n+1}. More precisely, suppose φ_{i+1} is consistent with the belief after observing $\varphi_1, \ldots, \varphi_i$ for $i \leq n$, but φ_{n+1} is inconsistent with the beliefs after observing $\varphi_1, \ldots, \varphi_n$. Let k be the maximal index such that φ_{n+1} is consistent with the beliefs after learning $\varphi_1, \ldots, \varphi_k$. The agent's beliefs after observing φ_{n+1} are the same as her beliefs after observing $\varphi_1, \ldots, \varphi_k, \varphi_{n+1}$. Thus, the agent acts as though she did not observe $\varphi_{k+1}, \ldots, \varphi_n$.

Boutilier does not provide any argument for the reasonableness of this ontology. In fact, Boutilier's presentation (like almost all others in the literature) is not in terms of an ontology at all; he presents natural revision as an attempt to minimize changes to the ranking. While the intuition of minimizing changes to the ranking seems reasonable at first, it becomes less reasonable when we realize its ontological implications. The following example, due to Darwiche and Pearl [1994], emphasizes this point. Suppose we encounter a strange new animal and it appears to be a bird, so we believe it is a bird. On closer inspection, we see that it is red, so we believe that it is a red bird. However, an expert then informs us that it is not a bird, but a mammal. Applying natural revision, we would no longer believe that the animal is red. This does not seem so reasonable.

One more point is worth observing: As described by Boutilier [1993], natural revision does not allow revision by *false*. While we could, of course, modify the definition to handle *false*, it is more natural simply to disallow it. This suggests that, whatever ontology is used to justify natural revision, in that ontology, revising by *false* should not make sense.

3.2 FREUND AND LEHMANN'S APPROACH

Freund and Lehmann [1994] stick close to the original AGM approach. They work with belief sets, not more general epistemic states. However, they are interested in iterated revision. They consider the effect of adding just one more postulate to the basic AGM postulates, namely

R9. If $\neg\varphi \in K$, then $K * \varphi = K_\perp * \varphi$,

where $K_\perp$ is the inconsistent belief set, which consists of all formulas.

Suppose $*$ satisfies K1–K9. Just as with Boutilier's natural revision, if φ_{i+1} is consistent with the beliefs after learning $\varphi_1, \ldots, \varphi_i$ for $i \leq n-1$, then $K * \varphi_1 * \ldots * \varphi_n = K * (\varphi_1 \wedge \ldots \wedge \varphi_n)$. However, if we then observe φ_{n+1}, and it is inconsistent with $K * \varphi_1 \wedge \ldots \wedge \varphi_n$, then $K * \varphi_1 * \ldots * \varphi_{n+1} = K_\perp * \varphi_{n+1}$. That is, observing something inconsistent causes us to retain none of our previous beliefs, but to start over from scratch. While the ontology here is quite simple to explain, as Freund and Lehmann themselves admit, it is a rather severe form of belief revision. Darwiche and Pearl's red bird example applies to this approach as well.

3.3 DARWICHE AND PEARL'S APPROACH

Darwiche and Pearl [1994] suggest a set of postulates extending the AGM postulates, and claim to provide a semantics that satisfies them. Their intuition is that the revision operator should retain as must as possible certain parts of the ordering among worlds in the ranking. In particular, if w

and w' both satisfy φ, then a revision by φ should not change the relative rank of w and w'. Similarly, if both w and w' satisfy $\neg\varphi$, then a revision should not change their relative rank. They describe four postulates that are meant to embody these intuitions:

C1. If $\varphi \vdash \psi$, then $(K * \psi) * \varphi = K * \varphi$
C2. If $\varphi \vdash \neg\psi$, then $(K * \psi) * \varphi = K * \varphi$
C3. If $\psi \in K * \varphi$, then $\psi \in (K * \psi) * \varphi$
C4. If $\neg\psi \notin K * \varphi$, then $\neg\psi \notin (K * \psi) * \varphi$

Freund and Lehmann [1994] point out that C2 is inconsistent with the AGM postulates. This observation seems inconsistent with the fact that Darwiche and Pearl claim to provide an example of a revision method that is consistent with their postulates. What is going on here? It turns out that the issues raised earlier help clarify the situation.

The semantics that Darwiche and Pearl use as an example is based on a special case of Spohn's *ordinal conditional functions* (OCFs) [1988] called *κ-rankings* [Goldszmidt and Pearl 1992]. A κ-ranking associates with each world either a natural number n or ∞, with the requirement that for at least one world w_0, we have $\kappa(w_0) = 0$. We can think of $\kappa(w)$ as the rank of w, or as denoting how surprising it would be to discover that w is the actual world. If $\kappa(w) = 0$, then world w is unsurprising; if $\kappa(w) = 1$, then w is somewhat surprising; if $\kappa(w) = 2$, then w is more surprising, and so on. If $\kappa(w) = \infty$, then w is impossible.[3] OCFs provide a way of ranking worlds that is closely related to, but has a little more structure than the orderings considered by Boutilier. The extra structure makes it easier to define a notion of conditioning.

Given a formula φ, let $\kappa(\varphi) = \min\{\kappa(w) : w \models \varphi\}$; we define $\kappa(\mathit{false}) = \infty$. We say that φ *is believed with firmness* $\alpha \geq 0$ *in OCF* κ if $\kappa(\varphi) = 0$ and $\kappa(\neg\varphi) = \alpha$. Thus, φ is believed with firmness α if φ is unsurprising and the least surprising world satisfying $\neg\varphi$ has rank α. We define $\mathrm{Bel}(\kappa)$ to consist of all formulas that are believed with firmness at least 1.

Spohn defined a notion of conditioning on OCFs. Given an OCF κ, a formula φ such that $\kappa(\varphi) < \infty$, and $\alpha \geq 0$, $\kappa_{\varphi,\alpha}$ is the unique OCF satisfying the property desired by Darwiche and Pearl—namely, if w and w' both satisfy φ or both satisfy $\neg\varphi$, then revision by φ should not change the relative rank of w and w', that is, $\kappa_{\varphi,\alpha}(w) - \kappa_{\varphi,\alpha}(w') = \kappa(w) - \kappa(w')$—such that φ is believed with firmness α in $\kappa_{\varphi,\alpha}$. It is defined as follows:

$$\kappa_{\varphi,\alpha}(w) = \begin{cases} \kappa(w) - \kappa(\varphi) & \text{if } w \text{ satisfies } \varphi \\ \kappa(w) - \kappa(\neg\varphi) + \alpha & \text{if } w \text{ satisfies } \neg\varphi. \end{cases}$$

Notice that $\kappa_{\varphi,\alpha}$ is defined only if $\kappa(\varphi) < \infty$, that is, if φ is considered possible.

Darwiche and Pearl defined the following revision function on OCFs:

$$\kappa *_{DP} \varphi = \begin{cases} \kappa & \text{if } \kappa(\neg\varphi) \geq 1 \\ \kappa_{\varphi,1} & \text{otherwise.} \end{cases}$$

Thus, if φ is already believed with firmness at least 1 in κ, then κ is unaffected by a revision by φ; otherwise, the effect of revision is to modify κ by conditioning so that φ ends up being believed with degree of firmness 1. Intuitively, this means that if φ is not believed in κ, in $\kappa * \varphi$ it is believed, but with the minimal degree of firmness.

It is not hard to show that if we take an agent's epistemic state to be represented by an OCF, then Darwiche and Pearl's semantics satisfies all the AGM postulates modified to apply to epistemic states (that is, R1′–R8′ in Section 2), except that revising by *false* is disallowed, just as in natural revision, so that R5′ holds vacuously; in addition, this semantics satisfies Darwiche and Pearl's C1–C4, modified to apply to epistemic states. For example, C2 becomes

C2′. If $\varphi \vdash \neg\psi$, then $\mathrm{Bel}((E * \psi) * \varphi) = \mathrm{Bel}(E * \varphi)$.

Indeed, as Darwiche and Pearl observe, natural revision also satisfies C1′–C4′, however, it has properties that they view as undesirable. Thus, Darwiche and Pearl's claim that their postulates are consistent with AGM is correct, if we think at the level of general epistemic states. On the other hand, Freund and Lehmann are quite right that R1–R8 and C1–C4 are incompatible; indeed, as they point out, R1–R4 and C2 are incompatible. The importance of making clear exactly whether we are considering the postulates with respect to the OCF κ or the belief set $\mathrm{Bel}(\kappa)$ is particularly apparent here.

The fact that Boutilier's natural revision also satisfies C1′–C4′ clearly shows that these postulates do not capture all of Darwiche and Pearl's intuitions. Their semantics embodies further assumptions. Some of them seem *ad hoc*. Why is it reasonable to believe φ with a *minimal* degree of firmness after revising by φ? Rather than trying

[3] Spohn allowed ranks to be arbitrary ordinals, not just natural numbers, and did not allow a rank of ∞, since, for philosophical reasons, he did not want to allow a world to be considered impossible. As we shall see, there are technical advantages to introducing a rank of ∞.

to come up with an improved collection of postulates (which Darwiche and Pearl themselves suggest might be a difficult task), it seems to us a more promising approach is to find an appropriate ontology.

3.4 LEHMANN'S REVISED APPROACH

Finally, we consider Lehmann's "revised" approach to belief revision [1995]. With each sequence σ of observations, Lehmann associates a belief set that we denote $\mathrm{Bel}(\sigma)$. Intuitively, we can think of $\mathrm{Bel}(\sigma)$ as describing the agent's beliefs after making the sequence σ of observations, starting from her initial epistemic state. Lehmann allows all possible sequences of consistent formulas. Thus, he assumes that the agent does not observe *false*. We view Lehmann's approach essentially as taking the agent's epistemic state to be the sequence of observations made, with the obvious revision operator that concatenate a new observation to the current epistemic state. The properties of belief change depend on the function Bel. Lehmann require Bel to satisfy the following postulates (where σ and ρ denote sequences of formulas, and $\cdot$ is the concatenation operator):

I1. $\mathrm{Bel}(\sigma)$ is a consistent belief set

I2. $\varphi \in \mathrm{Bel}(\sigma \cdot \varphi)$

I3. If $\psi \in \mathrm{Bel}(\sigma \cdot \varphi)$, then $\varphi \Rightarrow \psi \in \mathrm{Bel}(\sigma)$

I4. If $\varphi \in \mathrm{Bel}(\sigma)$, then $\mathrm{Bel}(\sigma \cdot \varphi \cdot \rho) = \mathrm{Bel}(\sigma \cdot \rho)$

I5. If $\psi \vdash \varphi$, then $\mathrm{Bel}(\sigma \cdot \varphi \cdot \psi \cdot \rho) = \mathrm{Bel}(\sigma \cdot \psi \cdot \rho)$

I6. If $\neg\psi \notin \mathrm{Bel}(\sigma \cdot \varphi)$, then $\mathrm{Bel}(\sigma \cdot \varphi \cdot \psi \cdot \rho) = \mathrm{Bel}(\sigma \cdot \varphi \cdot \varphi \wedge \psi \cdot \rho)$

I7. $\mathrm{Bel}(\sigma \cdot \neg\varphi \cdot \varphi) \subseteq \mathrm{Cl}(\mathrm{Bel}(\sigma) \cup \{\varphi\})$

We refer the interested reader to [Lehmann 1995] for the motivation for these postulates. As Lehmann notes, the spirit of the original AGM postulates is captured by these postulates. Lehmann views I5 and I7 as two main additions to the basic AGM postulates. He states that "Since postulates I5 and I7 seem secure, i.e., difficult to reject, the postulates I1–I7 may probably be considered as a reasonable formalization of the intuitions of AGM." Our view is that it is impossible to decide whether to accept or reject postulates such as I5 or I7 (or, for that matter, any of the other postulates) without an explicit ontology. There may be ontologies for which I5 and I7 are reasonable, and others for which they are not. "Reasonableness" is not an independently defined notion; it depends on the ontology. The ontology of the next section emphasizes this point.

4 TAKING OBSERVATIONS TO BE KNOWLEDGE

We now consider an ontology where observations are taken to be knowledge. As we said in the introduction, in this ontology, the agent has some (closed) set of formulas that he *knows* to be true, which is included in a larger set of formulas that he *believes* to be true. The belief set can be viewed as the result of applying some nonmonotonic reasoning system grounded in the observations. We can think of there being an ordering on the strength of his beliefs, with the formulas known to be true—the observations and their consequences—having the greatest strength of belief. Because observations are taken to be knowledge, any formula observed is added to the stock of knowledge (and must be consistent with what was previously known). In this ontology, it is impossible to observe *false*. In fact, it is impossible to make any inconsistent sequence of observations. That is, if $\varphi_1, \ldots, \varphi_n$ is observed, then $\varphi_1 \wedge \ldots \wedge \varphi_n$ must be consistent (although it may not be consistent with the agent's original beliefs).

In earlier work [1995a], we presented one way of formalizing this ontology, based on Halpern and Fagin's [1989] framework of multi-agent systems (see [Fagin, Halpern, Moses, and Vardi 1995] for more details). The key assumption in the multi-agent system framework is that we can characterize the system by describing it in terms of a *state* that changes over time. Formally, we assume that at each point in time, the agent is in some *local state*. Intuitively, this local state encodes the information the agent has observed thus far. There is also an *environment*, whose state encodes relevant aspects of the system that are not part of the agent's local state. A *global state* is a tuple (s_e, s_a) consisting of the environment state s_e and the local state s_a of the agent. A *run* of the system is a function from time (which, for ease of exposition, we assume ranges over the natural numbers) to global states. Thus, if r is a run, then $r(0), r(1), \ldots$ is a sequence of global states that, roughly speaking, is a complete description of what happens over time in one possible execution of the system. We take a *system* to consist of a set of runs. Intuitively, these runs describe all the possible behaviors of the system, that is, all the possible sequences of events that could occur in the system over time.

Given a system $\mathcal{R}$, we refer to a pair (r, m) consisting of a run $r \in \mathcal{R}$ and a time m as a *point*. If $r(m) = (s_e, s_a)$, we define $r_a(m) = s_a$ and $r_e(m) = s_e$. We say two points (r, m) and

(r', m') are *indistinguishable* to the agent, and write $(r, m) \sim_a (r', m')$, if $r_a(m) = r'_a(m')$, i.e., if the agent has the same local state at both points. Finally, an *interpreted system* is a tuple $(\mathcal{R}, \pi)$, consisting of a system $\mathcal{R}$ together with a mapping π that associates with each point a truth assignment to the primitive propositions.

To capture the AGM framework, we consider a special class of interpreted systems: We fix a propositional language $\mathcal{L}$. We assume that the agent makes observations, which are characterized by formulas in $\mathcal{L}$, and that her local state consists of the sequence of observations that she has made. We assume that the environment's local state describes which formulas are actually true in the world, so that it is a truth assignment to the formulas in $\mathcal{L}$. As observed by Katsuno and Mendelzon [1991a], the AGM postulates assume that the world is *static*; to capture this, we assume that the environment state does not change over time. Formally, we are interested in the unique interpreted system $(\mathcal{R}^{AGM}, \pi)$ that consists of all runs satisfying the following two assumptions for every point (r, m):

- The environment's state $r_e(m)$ is a truth assignment to the formulas in $\mathcal{L}$ that agrees with π at (r, m) (that is, $\pi(r, m) = r_e(m)$), and $r_e(m) = r_e(0)$.
- The agent's state $r_a(m)$ is a sequence of the form $\langle \varphi_1, \ldots, \varphi_m \rangle$, such that $\varphi_1 \wedge \ldots \wedge \varphi_m$ is true according to the truth assignment $r_e(m)$ and $r_a(m-1) = \langle \varphi_1, \ldots, \varphi_{m-1} \rangle$.

Notice that the form of the agent's state makes explicit an important implicit assumption: that the agent remembers all her previous observations.

In an interpreted system, we can talk about an agent's knowledge: the agent knows φ at a point (r, m) if φ holds in all points (r', m') such that $(r, m) \sim_a (r', m')$. It is easy to see that, according to this definition, if $r_a(m) = \langle \varphi_1, \ldots, \varphi_m \rangle$, then the agent knows $\varphi_1 \wedge \ldots \wedge \varphi_m$ at the point (r, m): the agent's observations are known to be true in this approach. We are interested in talking about the agent's beliefs as well as her knowledge. To allow this, we added a notion of *plausibility* to interpreted systems in [Friedman and Halpern 1995b]. We consider a variant of this approach here, using OCFs, since it makes it easier to relate our observations to Darwiche and Pearl's framework.

We assume that we start with an OCF κ on runs such that $\kappa(r) \neq \infty$ for any run r. Intuitively, κ represents our prior ranking on runs. Initially, no runs is viewed as impossible. We then associate, with each point (r, m), an OCF $\kappa^{(r,m)}$ on the runs. We define $\kappa^{(r,m)}$ by induction on m. We take $\kappa^{(r,0)} = \kappa$, and we take $\kappa^{(r,m+1)} = \kappa^{(r,m)}_{\varphi_{m+1}, \infty}$, where $r_a(m+1) = \langle \varphi_1, \ldots, \varphi_{m+1} \rangle$. Thus, $\kappa^{(r,m+1)}$ is the result of conditioning $\kappa^{(r,m)}$ on the last observation the agent made, giving it degree of firmness ∞. Thus, the agent is treating the observations as knowledge in a manner compatible with the semantics for knowledge in the interpreted system. Moreover, since observations are known, they are also believed.

As we show in [Friedman and Halpern 1995a], this framework satisfies the AGM postulates R1′–R8′, interpreted on epistemic states. (Here we take the agent's epistemic state at the point (r, m) to consist of $r_a(m)$ together with $\kappa^{(r,m)}$.) Moreover, it is easy to verify that the framework also satisfies Darwiche and Pearl's postulates (appropriately modified to apply to epistemic states), except that the contentious C2 is now vacuous, since it is illegal to revise by ψ and then φ if $\varphi \vdash \neg\psi$.

How does this framework compare to Lehmann's? Like Lehmann's, there is an explicit attempt to associate beliefs with a sequence of revisions. However, we have restricted the sequence of revisions, since we are treating observations as knowledge. It is easy to see that I1–I3 and I5–I7 hold in our framework. However, since we have restricted the sequence of observations allowed, some of these postulates are much weaker in our framework than in Lehmann's. In particular, I7 is satisfied vacuously, since we do not allow a sequence of the form $\sigma \cdot \neg\varphi \cdot \varphi$. On the other hand, I4 is not satisfied in our framework. Our discussion in the introduction suggests a counterexample. Suppose that initially, $\kappa(p \wedge q) = 0$, $\kappa(\neg p \wedge q) = 1$, $\kappa(p \wedge \neg q) = 2$, and $\kappa(\neg p \wedge \neg q) = 3$. Thus, initially the agent believes both p and q, but believes p with firmness 1 and q with firmness 2. If the agent then observes $\neg p \vee \neg q$, he will then believe q but not p. On the other hand, suppose the agent first observes p. He still believes both p and q, of course, but now p is believed with firmness ∞. That means if he then observes $\neg p \vee \neg q$, he will believe q, but not p, violating I4. However, a weaker variant of I4 does hold in our system: if the agent *knows* φ, then observing φ will not change her future beliefs.

5 DISCUSSION

The goal of this paper was to highlight what we see as some methodological problems in much of the literature on belief revision. There has been (in our opinion) too much attention paid to postulates, and not enough to the underlying ontology.

An ontology must make clear what the agent's epistemic state is, what types of observations the agent can make, the status of observations, and how the agent goes about revising the epistemic state.

We have (deliberately) not been very precise about what counts as an ontology, and clearly there are different levels of detail that one can provide. Although some papers on belief revision have attempted to provide something in the way of an ontology, the ontology has typically been insufficient detailed to verify the reasonableness of the postulates. For example, Gärdenfors [1988]—whose ontology is far better developed than most—takes belief sets to consist of formulas that are accepted, and revision to be by formulas that are accepted. As we have seen, unless belief sets or the revision operator contain additional information (such as epistemic importance or strengths of beliefs) this ontology will violate R1. On the other hand, unless we are careful about how we add such information, we may well violate some other axioms (such as R3 or R5). In any case, it is the job of the ontology to make completely clear such issues as whether observations are believed to be true or known to be true, and if they are believed, what the strength of belief is. This issue is particularly important if we have epistemic states like rankings that are richer than belief sets. If observations are believed, but not necessarily known, to be true, then it is not clear how to go about revising such a richer epistemic state. With what degree of firmness should the new belief be held? No particular answer seems to us that well motivated. It may be appropriate for the user to attach degrees of firmness to observations, as was done in [Goldszmidt 1992; Williams 1994; Wobcke 1995] (following the lead of Spohn [1988]); we can even generalize to allowing uncertain observations [Dubois and Prade 1992].

In this paper we have focused on belief revision. However, the need to clarify the underlying ontology goes far beyond belief revision. Much the same comments can be made for all the work on nonmonotonic logic as well (this point is essentially made in [Halpern 1993]). Not surprisingly, our critique applies to other approaches belief change as well, and in particular to Katsuno and Mendelzon's *belief update* [1991a]. Although the motivation described by Katsuno and Mendelzon is different than that of revision, the discussion of update is stated in terms of postulates about belief states. Thus, for example, the distinction between the agent's belief state and epistemic state arises in update as well: update is defined as function from (belief states × formulas) to belief states. However, the agent's belief state does not uniquely determine the outcome of update. This is demonstrated, for example, by Katsuno and Mendelzon's semantic characterization that requires the agent to have a ternary relation on possible worlds such that $w_1 <_w w_2$ holds if the agent considers w_1 to be "closer" to w than w_2. Other issues we raise here also apply to update for similar reasons.

The ontology we propose in Section 4, where observations are treated as knowledge, can be applied to update as well. Moreover, the assumption that observations are true is less problematic for iterated update: since update does not assume that propositions are static, we can consider runs where φ is true at one time point, and false at the next one. Thus, assuming that observations are known to be true does not rule out sequences of observations of the form $\varphi, \neg\varphi, \ldots$. In fact, all sequences of consistent observations are allowed. We refer the reader to [Friedman and Halpern 1995a] for more details.

It seems to us that many of the intuitions that researchers in the area have are motivated by thinking in terms of observations as known, even if this is not always reflected in the postulates considered. We have examined carefully one particular instantiation of this ontology, that of treating observations as knowledge. We have shown that, in this ontology, some postulates that seem reasonable, such as Lehmann's I4, do not hold. We do not mean to suggest that I4 is "wrong" (whatever that might mean in this context). Rather, it shows that we cannot blithely accept postulates without making the underlying ontology clear. We would encourage the investigation of other ontologies for belief change.

Acknowledgements

The authors are grateful to Craig Boutilier, Adnan Darwiche, Adam Grove, and Daniel Lehmann for comments on a draft of this paper and useful discussions relating to this work. This work was supported in part by NSF Grant IRI-95-03109. The first author was also supported in part by IBM Graduate Fellowship.

References

Alchourrón, C. E., P. Gärdenfors, and D. Makinson (1985). On the logic of theory change: partial meet functions for contraction and revision. *J. Symbolic Logic 50*, 510–530.

Boutilier, C. (1992). Normative, subjective and autoepistemic defaults: adopting the Ramsey test. In *Principles of Knowledge Representation and Reasoning: Proc. 3rd Int. Conf. (KR '92)*, pp. 685–696.

Boutilier, C. (1993). Revision sequences and nested conditionals. In *Proc. 13th Int. Joint Conf. on Artificial Intelligence (IJCAI '93)*, pp. 519–525.

Boutilier, C. (1994). Unifying default reasoning and belief revision in a modal framework. *Artificial Intelligence 68*, 33–85.

Boutilier, C. and M. Goldszmidt (1993). Revising by conditional beliefs. In *Proc. Nat. Conf. on Artificial Intelligence (AAAI '93)*, pp. 648–654.

Darwiche, A. and J. Pearl (1994). On the logic of iterated belief revision. In *Theoretical Aspects of Reasoning about Knowledge: Proc. 5th Conf.*, pp. 5–23.

Dubois, D. and H. Prade (1992). Belief change and possibility theory. In P. Gärdenfors (Ed.), *Belief Revision*. Cambridge University Press.

Fagin, R., J. Y. Halpern, Y. Moses, and M. Y. Vardi (1995). *Reasoning about Knowledge*. MIT Press.

Freund, M. and D. Lehmann (1994). Belief revision and rational inference. Technical Report TR 94-16, Hebrew University.

Friedman, N. and J. Y. Halpern (1995b). Modeling belief in dynamic systems. part I: foundations. Technical Report RJ 9965, IBM. Submitted for publication. A preliminary version appears in R. Fagin editor. *Theoretical Aspects of Reasoning about Knowledge: Proc. 5th Conf.*, 1994, pp. 44–64, under the title "A knowledge-based framework for belief change. Part I: foundations".

Friedman, N. and J. Y. Halpern (1995a). Modeling belief in dynamic systems. Part II: revision and update. In preparation. A preliminary version appears in *Principles of Knowledge Representation and Reasoning: Proc. 4th Int. Conf. (KR '94)*, 1994, pp. 190–201, under the title "A knowledge-based framework for belief change. Part II: revision and update.".

Gärdenfors, P. (1988). *Knowledge in Flux*. MIT Press.

Gärdenfors, P. and D. Makinson (1988). Revisions of knowledge systems using epistemic entrenchment. In *Proc. 2nd Conf. on Theoretical Aspects of Reasoning about Knowledge*, pp. 83–95.

Goldszmidt, M. (1992). *Qualitative probabilities: a normative framework for commonsense reasoning*. Ph. D. thesis, University of California Los Angeles.

Goldszmidt, M. and J. Pearl (1992). Rank-based systems: A simple approach to belief revision, belief update and reasoning about evidence and actions. In *Principles of Knowledge Representation and Reasoning: Proc. 3rd Int. Conf. (KR '92)*, pp. 661–672.

Grove, A. (1988). Two modelings for theory change. *J. Philosophical Logic 17*, 157–170.

Halpern, J. Y. (1993). A critical reexamination of default logic, autoepistemic logic, and only knowing. In *Proc. 3rd Kurt Gödel Colloquium*, pp. 43–60.

Halpern, J. Y. and R. Fagin (1989). Modelling knowledge and action in distributed systems. *Distributed Computing 3*(4), 159–179.

Katsuno, H. and A. Mendelzon (1991a). On the difference between updating a knowledge base and revising it. In *Principles of Knowledge Representation and Reasoning: Proc. 2nd Int. Conf. (KR '91)*, pp. 387–394.

Katsuno, H. and A. Mendelzon (1991b). Propositional knowledge base revision and minimal change. *Artificial Intelligence 52*(3), 263–294.

Lehmann, D. (1995). Belief revision, revised. In *Proc. 14th Int. Joint Conf. on Artificial Intelligence (IJCAI '95)*, pp. 1534–1540.

Levi, I. (1988). Iteration of conditionals and the Ramsey test. *Synthese 76*, 49–81.

Rott, H. (1989). Conditionals and theory change: revision, expansions, and additions. *Synthese 81*, 91–113.

Shafer, G. (1976). *A Mathematical Theory of Evidence*. Princeton University Press.

Spohn, W. (1988). Ordinal conditional functions: a dynamic theory of epistemic states. In W. Harper and B. Skyrms (Eds.), *Causation in Decision, Belief Change and Statistics*, Vol. 2, pp. 105–134. Reidel.

Williams, M. (1994). Transmutations of knowledge systems. In *Principles of Knowledge Representation and Reasoning: Proc. 4th Int. Conf. (KR '94)*, pp. 619–629.

Wobcke, W. (1995). Belief revision, conditional logic, and nonmonotonic reasoning. *Notre Dame J. of Formal Logic 36*(1), 55–102.

Modeling Belief Change using Counterfactuals

Tom Costello*
Computer Science Department
Stanford University
Stanford, CA 94305
email: costello@steam.Stanford.EDU

Abstract

We can represent how to change our beliefs in the light of new information by using a conditional "if A is the case, then we should accept B". We propose that this belief change conditional should be defined as material implication when A is consistent with our current beliefs, and as a suitable counterfactual in the other case. We show that the resulting belief change system obeys the rationality postulates suggested by Alchourrón, Gärdenfors, and Makinson. To get an exact correspondence between systems of belief change thus defined in terms of counterfactuals, and systems based on rationality postulates, it is more intuitive to modify the usual "closest possible world" models of counterfactuals, so that different similarity measures centered at the same world are cotenable, and all worlds are ranked. Once this is done we can achieve a correspondence between finite sequences of belief changes using rationality postulates and models of a counterfactual.

The valid counterfactual sentences do not constitute a possible *state of belief*. They do not decide enough sentences. However, we can specify a state of belief, by adding the additional sentences, that describe the fact that we believe no counterfactuals, save those that we have explicitly been told. This characterizes a "tabula rasa" or blank slate, upon which we can impose any belief change system, by revising it with the sentences that characterize the defaults of that belief change system.

* The author thanks Anna Patterson, John McCarthy, Grigori Mints, and Johan van Benthem for helpful discussions.

1 Motivations

Changing ones mind is the prerogative of every fallible reasoner. We are constantly told new information that we must assimilate. That information can neatly be split into two parts; new facts that contradict our current beliefs, and facts that we currently have no opinion about.

There is considerable agreement about how to deal with the second case. We should conjoin the new information to our current beliefs. The first case is more problematic however. Clearly conjoining the information will lead to inconsistency. One feels tempted to assent to the new belief, but one does not want to give up all ones old beliefs.

A solution to this quandary might be found in counterfactuals. They are propositions of the form,"if ... were the case, then ... would be the case.", where the left hand side is contrary to our current beliefs.

There are several semantics that state what counterfactuals are true based only on which facts about the world are true, and some notion of what other worlds might be most likely. Perhaps the best known of these is the "official logic" for counterfactual conditionals, Lewis's **VC**. We intend to use counterfactuals to tell us what facts in our present corpus we should retain. However, we differ from Lewis in that we consider only what worlds are most likely when giving truth conditions to our counterfactual. We do not use what facts are true in the actual world.

Every belief change conditional $A > B$ can be divided into two parts—the material conditional, which is appropriate when we do not believe $\neg A$, and the counterfactual part, which is appropriate when we believe $\neg A$. These parts can define the *belief change conditional*.

If $A > B$ is the belief change conditional, we can relate

our counterfactual conditional $\succ$ to $>$[1] by the following identity.

$$\textbf{(Change)} \quad \Gamma \models A > B \stackrel{def}{\equiv}$$

$$\text{if } \Gamma \models \neg A \text{ then } \Gamma \models A \succ B \text{ else } \Gamma \models A \rightarrow B$$

This approach to conditionals allows a very simple formalization of how beliefs change in the light of new information. It avoids the problems, exposed by Lewis and Gärdenfors[Lewis, 1976, Lewis, 1991, Gärdenfors, 1986, Gärdenfors, 1988], that arise when belief change conditional is added to the language as an operator.

Adding a new connective to the language also avoids the problems exposed by Lehmann[Lehmann, 1995]. He points out that there are strong arguments against treating belief states, that have exactly the same sentences true in them, identically. He argues the change should also depend on the pedigree of the beliefs. As we represent more information, by having an extra connective $\succ$, we can have theories with the same $\succ$-free sentences, that are not identical.

2 Outline of the Paper

We wish to develop a model of belief change, based on the idea that we can use counterfactual conditionals to model how we change our mind when we find we are mistaken. When we find we are under-informed, we intend to conjoin the new information. To carry out this plan we need a model of counterfactuals, and a model of belief change.

We first look at a model of counterfactuals proposed by Lewis. It is based around the idea that a counterfactual $A \succ B$ is true if the corresponding implication $A \rightarrow B$ is true at the closest worlds where A is true. As a model of counterfactuals this system is not completely natural as it does not allow us to consider different similarity measures on the same world—each world must have a unique ordering on worlds assigned to it. It is natural to imagine that there are different possible orders of the same set of worlds. Furthermore this model makes the counterfactual $A \succ B$ agree with the material conditional when A is the case. Thus it gives truth conditions to this conditional, when it is not counterfactual.

The next model we look at seem superficially very similar to Lewis's. It is Grove's model [Grove, 1988] of the belief change postulates proposed by Alchourrón, Gärdenfors, and Makinson (AGM) [Alchourrón et al., 1985]. It is intended to deal with incomplete knowledge. Sadly, as shown by Gärdenfors[Gärdenfors, 1986, Gärdenfors, 1988] it cannot represent conditionals in its language, if the conditionals are to represent the notion of belief change itself.

Our solution to this problem is to propose a new model, that allows both nested counterfactual conditionals and incomplete knowledge. It combines some of the features of both models. Its most critical idea is that the underlying propositions that are believed are not objective propositions and the conditionals corresponding to belief change, but rather, objective propositions, and counterfactual conditionals. The belief change conditional is definable from the counterfactual conditional and objective propositions, but is not a "first class" belief. The critical change is that we only preserve the counterfactual conditional and the objective propositions. To stress the importance of this decision we list this as the assumption $\succ$.

($\succ$) Our language contains $\succ$ as its only non truth functional connective.

We present a new model of counterfactuals and show that, when **(Change)** is used to specify belief change in terms of this counterfactual, we get a system that obeys each of AGM's eight postulates. We thus suggest a new postulate, essentially the counterfactual part of **Change** that relates belief change to counterfactuals. This postulate completes the rationality postulates, in that it completely determines revision in terms of the logical connectives and $\succ$. We give theorems relating our postulates to our semantics. Namely, every set of iterated AGM revisions can be captured in our model. Thus we show a correspondence between rational belief change as defined by postulates and using **(Change)** to define belief change in terms of counterfactuals.

Finally we consider what state of belief is most appropriate to choose as an initial state. We show that we can define a set of sentences that capture the idea that initially we should believe as few counterfactuals as possible.

3 Previous Work

3.1 Lewis

Lewis suggested a semantics of conditionals based on a system of spheres. The basic notion is a sphere of

[1] Throughout the paper, we use $>$ to refer to a belief change conditional, and $\succ$ to refer to a counterfactual conditional.

accessibility, or alternately a set of possible worlds. To each possible world i he assigned a set of spheres of accessibility $\mathbb{S}_i$[2]. This assignment is constrained to meet certain formal constraints.

- **C** Centering $\{i\} \in \mathcal{S}_i$.
- **1** Nesting: $\forall S \in \mathcal{S}_i, T \in \mathcal{S}_i . T \subset S \vee S \subset T$.

This system of spheres carries the notion of similarity of worlds. Each sphere around a world i contains exactly those worlds that resemble i to a certain degree. We write that a proposition A is true in a world w as $w \models A$. We can now define Lewis's counterfactual conditional as:

$i \models_{Lewis} A \succ B \stackrel{def}{=}$
$\forall S \in \mathcal{S}_i, w \in S . w \not\models A$ or
$\exists S \in \mathcal{S}_i, w \in S . w \models A$ and $\forall w' \in S . w' \models A \rightarrow B$

The first alternative gives the vacuous case where there is no world where A is possible, the second states that we judge $A \succ B$ to be true if in the closest worlds where A is true, B is true.

Thus Lewis suggested that we could judge conditionals by appealing to what is essentially a "well-founded"[3] total pre-order on worlds. We can see that the system of spheres induces a total pre-order, where $w \leq w'$ iff $\exists S . w \in S \wedge w' \notin S$.

Thus Lewis's system is designed to capture that part of our beliefs that will be useful when we commit sins of commission, when we find a new fact that contradicts our old beliefs.

Lewis's system can be axiomatized in the following way, as he originally proposed,

$$\frac{A, \quad A \rightarrow B}{B}$$

all truth functional tautologies

$$\frac{C_1 \wedge \ldots \wedge C_n \rightarrow B}{((A \succ C_1) \wedge \ldots \wedge (A \succ C_n)) \rightarrow (A \succ B)} \text{ for } n \geq 1$$

$A \succ A$
$(\neg A \succ A) \rightarrow (B \succ A)$
$(A \succ \neg B) \vee (((A \wedge B) \succ C) \equiv (A \succ (B \rightarrow C)))$

$(A \succ B) \rightarrow (A \rightarrow B)$
$(A \wedge B) \rightarrow (A \succ B)$

or in the manner of Gärdenfors[Gärdenfors, 1978], to mirror revision style semantics, or using the notation of non-monotonic consequence relations which we now present.

All valid principles of propositional logic
Replacement of Provable Equivalents

$$A \succ A \qquad \text{Reflexitivity}$$

$$\frac{A \succ B}{A \succ (B \vee C)} \qquad \text{Right Monotonicity}$$

$$\frac{A \succ B, \; A \succ C}{A \succ (B \wedge C)} \qquad \text{Conjunction}$$

$$\frac{A \succ B, \; C \succ B}{(A \vee C) \succ B} \qquad \text{Disjunction}$$

$$\frac{A \succ B, \; A \succ C}{(A \wedge B) \succ C} \qquad \text{Cautious Monotonicity}$$

$$\frac{A \succ B, \; \neg(A \succ C)}{(A \wedge \neg C) \succ B} \qquad \text{Rationality}$$

$$\frac{True \succ A}{A} \qquad \text{Centering}$$

These systems are equivalent. The second is equivalent as shown by Burgess[Burgess, 1981]. We give the second axiomatization, as it will bear much similarity to the way in which we axiomatize our conditional, and shows how similar counterfactual reasoning is to work in non-monotonic consequence relations[Kraus et al., 1990].

It is worth noting that although Lewis's system is designed to deal with counterfactual conditionals, it also gives truth conditions for conditional sentences that are not counterfactual, that is sentences whose premise is not counterfactual. In this case, Lewis gives the conditional the same truth value as the material conditional. This is seen in his axiom, $(A \succ B) \rightarrow (A \rightarrow B)$. This has a consequence, $(True \succ A) \rightarrow A$, the assumption of centering.

3.2 Grove

Grove suggested a semantics for the AGM postulates for belief revision. He characterized the conditional "if I revised my beliefs to accommodate A, the I would would believe B". He also uses a system of spheres, but his system differs from Lewis's. Rather than define what conditionals are true at a world, he starts by

[2] We use A, B, C, D for propositions or sentences. We use w, i, j, k for worlds, which we consider set of propositions. We use K, M, N, Γ, Δ, Θ for sets of sentences. We use L for languages. We use W, R, S for sets of worlds, and calligraphic letters for sets of sets of worlds, $\mathcal{S}$, $\mathcal{W}$, $\mathcal{R}$, $\mathcal{S}$. We use $\mathbb{S}$ for rich states of belief. We use P for propositional letters, and $\mathcal{P}$ for sets of propositional letters. We use $\mathcal{K}$ for sets of sets of sentences.

[3] The condition is not exactly well-foundedness, but Kraus et al.'s [Kraus et al., 1990] smoothness.

defining what conditionals are true at a set of worlds. We note that he implicitly rejects the idea that one can determine what conditionals are true at a set of worlds by looking at the conditionals true at each world in that set,

We write $[A]$ for the set of worlds that satisfy a proposition A. His spheres are also sets of possible worlds. There is a system of spheres corresponding not to each world, but to each set of worlds. Thus instead of the system of spheres from a world i we have the system of spheres around a set of worlds $[K]$. The smallest sphere is the sphere $[K]$, that is the set of worlds itself. Contrast this with Lewis who has a systems of spheres associated with a single world. The largest sphere is all the worlds. For every sentence A, the set of spheres around a set $[K]$ that contain worlds that satisfy A has a minimum, in the sense of set inclusion. We call this minimum S_A.

Grove's interpretation of the conditional is, for a system of spheres, centered on the set [K],

$$K \models_{Grove} A > B \stackrel{def}{=} B \in \bigcap_{w \in [A] \cap S_A} w$$

We see that the conditional is defined as B being true in the smallest worlds where A is true. Grove's system tells us how to add to our knowledge. If A is not the case in $[K]$, but $[K] \cap [A]$ is non-empty then we find ourselves at the set of worlds $[K] \cap [A]$. In the case that $[K] \cap [A]$ is empty, then we find ourselves in a system very much like Lewis's. The key insight for us, is that we can represent our partial knowledge of the world by a set of possible worlds, and that we can represent revision by selecting out the subset of these worlds where the new fact is the case, when this subset is non-empty.

It is worth noting that the Grove semantics do not give a semantics to nested conditionals. We might imagine that we could use the set of worlds picked out by S_A, to give a new set of worlds from which to further judge the truth of sub-formulas. Grove does not do this because his intent was to capture a set of postulates, proposed by AGM, that render trivial extending the semantics in this way.

3.3 Nesting

There has been some work that considers nested conditionals. Adams [Adams, 1975] has suggested that $A > (B > C)$ might be read as $A \wedge B > C$, under suitable restrictions.

Lehman[Lehmann, 1995] suggests seven postulates that both strengthen and weaken, in some ways, the AGM postulates, by adding Darwiche and Pearl's[Darwiche and Pearl, 1994] **C1**, and a weaker version of their **C2**, which Freund and Lehmann[Freund and Lehmann, 1994] showed was inconsistent. He weakens the postulates by cropping the implicit assumption that revision is functional on theories. This work has a close connection with non-monotonic consequence relations. In this system revising one's beliefs by A is equivalent to adding the weakest sentence that non-monotonically implies A if A is consistent with the current beliefs. Note that this is consistent with the current monotonic beliefs, not with the non-monotonic consequences of those beliefs.

4 Rationality Postulates

Alchourrón, Gärdenfors, and Makinson give eight rationality postulates that they claim every belief change system should obey. Before we can speak of these postulates we must first introduce some notation.

Definition: 1 (Language) *A language L is identified with a set of sentences. We will consider languages that contain all the truth functional connectives, $\{\wedge, \vee, \rightarrow, \neg, \equiv\}$, a set of propositional atoms $\mathcal{P}$, a conditional connective $\succ$, and the additional constants $\perp$ and T. The rules for constructing sentences are the standard ones. We call this $L^{\succ}_{\mathcal{P}}$. We call the language without $\succ$, $L^{classical}_{\mathcal{P}}$*

Definition: 2 (State of Belief) *A state of belief K_L over a language L, is a logically closed subset of the sentences of the language L.*

We use $Cn(\{A_1, \ldots, A_n\})$ to refer to the logical closure of the set of sentence $\{A_1, \ldots, A_n\}$. We use $K+A$ to refer to the logical closure of the set $K \cup \{A\}$.

Definition: 3 (Revision) *We define the revision of a state of belief K_L, by a sentence A in L, as $K_L * A$. When the language is fixed we drop the subscript L.*

$K * A$ represents the state of belief which is the result of minimally changing K to include A.

We can now define a *belief revision system.*

Definition: 4 (Belief Revision System) *A belief revision system, $\langle \mathcal{K}, * \rangle$, is a sets of states of beliefs, over a language L, and a function $*$ from pairs of states of belief and sentences, to states of beliefs $* : \mathcal{K} \times L \rightarrow \mathcal{K}$.*

We can now state the eight postulates proposed by AGM:

($R1$) $K * A$ is deductively closed.
($R2$) $A \in K * A$.
($R3$) $K * A \subseteq K + A$.
($R4$) If $\neg A \notin K$ then $K + A \subseteq K * A$.
($R5$) $K * A = Cn(\bot)$ if and only if $\models \neg A$.
($R6$) If $\models A \equiv B$ then $K * A = K * B$.
($R7$) $K * (A \wedge B) \subseteq (K * A) + B$.
($R8$) If $\neg B \notin K * A$ then $(K * A) + B \subseteq K * (A \wedge B)$.

These postulates do not fully constrain belief change. Indeed, we need to add extra information, in the form of a choice of sphere around each set of worlds in Grove's model, or an entrenchment ordering from Gärdenfors [Gärdenfors, 1988], or a partial meet contraction function [Alchourrón et al., 1985], to characterize a belief change operator exactly. Rather than add one of these ordering principles outside the language, we propose to add a new rationality postulate:

(Counter)
If $\neg A \in K$ then $B \in K * A$ if and only if $A \succ B \in K$.

This will uniquely give a belief change system. We show this in Theorem 3. However to make sense of this postulate we need a semantics for the counterfactual $\succ$.

The idea behind **(Counter)** is that when we believe $\neg A$, we should change our beliefs as we would when we consider a counterfactual. In any other case, ($R3$) and ($R4$) will apply and fully describe the revision.

This approach, of using a conditional to define belief change seems very close to work done by Gärdenfors in [Gärdenfors, 1988]. The major difference is that we only characterize belief change using the conditional when the conditional is truly a counterfactual, i.e. when ($R4$) does not apply.

5 Expanding Counterfactual Semantics

We wish to define a belief revision conditional in terms of counterfactuals. However when we attempt to do this we see two problems in Lewis's standard model. The counterfactuals that are true are indexed only by a world. We wish to have our semantics capture states of belief, and thus identify worlds and sets of propositions.

If we take a commonly considered counterfactual:

> If Otto had come to the party, it would have been a good party.

Some people may consider this true, and others false. They may differ on their beliefs, not because the objective facts they know about reality differ, but instead because one person considers the world where Otto came and the party was good, closer than the world where Otto came and the party was bad, and the other person does not. They can differ on what they consider the ranking on worlds, without differing on what they believe the current world to be.

Lewis's model insists that each world have a unique ranking associated with it. He speaks of the system of spheres associated with each world. We can generalize his semantics by allowing the same world to have different system of spheres around it. Once we do this it is clear that the system of spheres should rank pairs of worlds and systems of spheres, not worlds.

Lewis did not include this freedom in his semantics, but it makes very little difference, as the semantics allow arbitrary many copies of worlds that agree on all propositions.

We wish to be able to work out, from a "state of belief", what counterfactuals that reasoner should believe. We thus want to have as our objects states of belief, rather than worlds. Unless the state of belief includes some representation of the similarity measure it uses, we will not be able to calculate what counterfactuals should be accepted. To represent the state of belief that results after being told a previously denied proposition, as a set of conditionals, the the truth conditions of the counterfactual must bring us not to a new world, but to a full state of belief, that is, a world and a new similarity measure on state of belief. For this reason we need the primary index in the satisfaction relation, or the *state of belief*, to be a set of assignments to objective propositions, and an order on states of belief.

The second difference between our system and Lewis's is that we do not care what the truth conditions of counterfactuals are when their premises are not contradictory with what we currently believe. This might seem like a minor point, however it allows us to simplify the system somewhat. In particular, we lose the axiom corresponding to centering. This is what allows us to avoid the unnecessary updating of conditionals, when we revise our beliefs by accepting consistent new information.

The third difference between our semantics and Lewis's is that we wish that the conditional $A \succ \bot$ be true only when A is contradictory. Thus every state of belief smoothly orders all other states of belief.

6 Our Semantics

Our basic unit is a *rich* state of belief $\langle W, \leq \rangle$, a set of worlds W, which capture what sentences in the object language we consider possible, and a total order on rich states of belief[4]. Our total order plays the same role as Lewis spheres. This is the semantical notion that parallels the syntactic states of belief we saw earlier. We note that we use a total order, rather than a total pre-order. This difference will be dealt with in the definition of the satisfaction relation.

Definition: 5 (Rich State of Belief) *A rich state of belief, $\mathbb{S}$ over a set of propositional atoms $\mathcal{P}$, is a pair $\langle W, \leq_{\mathcal{P}} \rangle$, a set of worlds W, identified with subsets of $\mathcal{P}$, and a smooth[5] total order on rich states of belief over $\mathcal{P}$.*

We write the set of all rich states of belief over $\mathcal{P}$ as $\mathcal{W}_{\mathcal{P}}$.

It should be stressed that our semantics agree with Lewis's semantics on all conditionals, save that we add that a proposition conditionally implies false only when it is logically equivalent to false as we wish to avoid contradiction for as long as possible.

Definition: 6 ($\downarrow_{\leq} A$) *If $\mathbb{S} = \langle W, \leq \rangle$, is a rich state of belief, and $W' \subset W$, $W' \neq \emptyset$ then we say that $\langle W', \leq \rangle$ is a sub-state of $\mathbb{S}$.*

We define $\downarrow^{*}_{\leq} A$ to be the smallest state under the pre-order $\leq$, that has a sub-state that models A.

We define $\downarrow_{\leq} A$ to be the sub-state of $\downarrow^{*}_{\leq} A$ that models A, and whose set of worlds is maximal among the sub-states of $\downarrow^{*}_{\leq} A$ that model A.

We now define satisfaction or $\models_T$ a relation between rich states of beliefs and propositions. We now inductively define the satisfaction relation between rich sates of belief and formulas.

[4]Our semantics of conditional are best explained in terms of non-well-founded sets. As our model of non-well-founded sets we take ZF with urelements, with AAFAA, that is Aczel's Anti-foundation Axiom with Atoms. This states that there is a set associated with every labeled directed graph. It is the use of non-well-founded sets that allows this recursive definition. Rather than use non-well founded set theory we could use any of the techniques for constructing models of the untyped lambda calculus[Scott, 1972, Plotkin, 1975]. Our treatment is similar to von Rimsha's[von Rimscha, 1980] non-well founded model of the untyped lambda calculus.

[5]We mean smooth in Kraus et al.'s sense, that is that every definable subset has a least element.

Definition: 7 ($\langle W, \leq \rangle \models_T A$) *We now define the satisfaction relation for our system $\models_T$. It is a relation between a rich state of belief in $\mathcal{W}_{\mathcal{P}}$, and a sentence in the language $L^{\succ}_{\mathcal{P}}$. We define it using structural induction over the language $L^{\succ}_{\mathcal{P}}$.*

- $\langle W, \leq \rangle \models_T P$ *if and only if* $\forall w \in W. w \models_T P$.
- $\langle W, \leq \rangle \models_T \neg A$ *if and only if*, $\forall w \in W. \langle \{w\}, \leq \rangle \not\models_T A$.
- $\langle W, \leq \rangle \models_T A \wedge B$ *if and only if* $\langle W, \leq \rangle \models_T A$ *and* $\langle W, \leq \rangle \models_T B$.
- $\langle W, \leq \rangle \models_T A \succ B$ *if and only if* $\forall W', \leq' . \langle W', \leq' \rangle \models_T \neg A$ *or* $\downarrow_{\leq} A \models_T B$

This gives us a notion of truth at a rich state of belief.

7 Completeness

We can axiomatize our system, using the axioms of **V**, that is the axioms of **VC** less centering, and the axiom that a proposition counterfactually entails false, only when it is itself contradictory. We denote provability in this system as $\vdash_T$. It axioms are the axioms of **V**, plus,

$$A \succ \bot \rightarrow \vdash_T \neg A.$$

We note that this notion of derivation is recursively defined in the sense that its axioms refer to its own notion of derivation. We therefore can have more than one fixed point for the set of theorems. We take the smallest fix-point as our definition, that is, a sentence is only derivable if it is derivable in every fix-point.

We now show the completeness of $\vdash_T$ for our semantics. We follow the usual method of making the objects of our model from sets of sentences. The sets of sentences that we associate with rich states of beliefs are those sets Δ, of the form $\Delta = \{A | B \succ A \in \Gamma\}$, for some maximally consistent set of sentences Γ, and sentence B. We use Γ to refer to a maximally consistent set of sentences, and Δ to refer to a set of the form $\{A | B \succ A \in \Gamma\}$. We call sets of the form $\{A | B \succ A \in \Gamma\}$, for Γ maximally consistent, *pre-maximal* sets.

We associate a rich state of belief S with a pre-maximal set of sentences Δ in the following way. We take the valuations of S to be the worlds that model those conditional free sentences A, such that $A \in \Delta$. We note that this differs from the usual method of associating a valuation with state of belief. More normally we would

associate the set of letters the state of belief contained as its valuation. The reason we do not do this is because we are interested in having states of belief that have more that one valuation.

The proof goes by a series of lemmas. We first show that there is a unique non-empty set of worlds associated with each pre-maximal set of sentences Δ. We next show that we can make a total order on rich states of belief from the order we read off. Next we show that all the sentences in a pre-maximal Δ are true in some rich state of belief made from Δ. This is the model for Δ.

We define a pre-state of belief to be a set of worlds and a binary relation in pre-state of belief.

Definition: 8 *We define the function $\mathcal{S}(\Delta)$ from pre-maximal sets of formulas Δ to pre-states of belief as follows. We let $\mathcal{S}(\Delta) = \langle W, \leq \rangle$, then*

- *W is the largest set of worlds such that if $W \models B$, B conditional free, then $B \in \Delta$.*
- *If $A \vee B \succ A \in \Delta$, then $\mathcal{S}\{\alpha | A \succ \alpha \in \Delta\} \leq \mathcal{S}\{\beta | B \succ \beta \in \Delta\}$.*

Lemma: 1 *If $\mathcal{S}(\Delta) = \langle W, \leq \rangle$, for some consistent pre-maximal set of sentences Δ, then there is a unique maximal W.*

Proof: We consider the conditional free, non-refutable sentences B such that $B \in \Gamma$. It is logically closed, as the logic is closed under substitution by equivalents. Thus it is the theory of a set of worlds. Clearly there is no larger set with this property.□

Lemma: 2 *Given a pre-maximal consistent set of sentences Δ, then the relation, $A \leq B$ as,*

$$A \leq B \text{ iff } A \vee B \succ A \in \Delta$$

is a total order on sets of logically equivalent sentences.

Proof: We need to show this is transitive, comparable and reflexive. Reflexitivity follows from right weakening and left logical equivalence, comparability from cumulativity, Or, and rationality, and transitivity follows from Or, Left Logical Equivalence, Right Weakening and cautious monotonicity. □.

Lemma: 3 *The function $\mathcal{S}$ can be extended to a function from pre-maximal sets of sentences to rich states of belief.*

Proof: We need to show that $\mathcal{S}(\Delta)$, for a consistent set of sentences Δ, can be extended to a rich state of belief. To do this we note that it is already a set of worlds and a total order on pre-states of belief. We thus need to extend the total order to all rich states of belief. We can do this by choosing a smooth total order on rich states of belief and using this order to order all rich states of beliefs not ordered by $\mathcal{S}$. □

We call this new function to rich state of beliefs $\mathbf{S}(\Gamma)$.

Lemma: 4 *For all $A \in \Delta,$, Δ pre-maximal, $\mathbf{S}(\Delta) \models A$*

Proof: We show this by induction on the structure of A.

- $A = p$. Immediate from definition of $\mathbf{S}(\Gamma)$.
- $A = B \wedge C$. Immediate from the definition of $\models$, and the induction hypothesis.
- $A = B \vee C$.Immediate from the definition of $\models$, and the induction hypothesis.
- $A = \neg B$. Immediate from the definition of $\models$, and the induction hypothesis.
- $A = B \succ C$. We need to show that in the minimal rich state of belief where B is true C is the case. We this need to show that that C is true in $(\{A | B \succ A \in \Delta\})$. But, by assumption, $B \succ C \in \Delta$, thus from the induction hypothesis, C is true in, $\mathbf{S}(\{A | B \succ A \in \Delta\})$, as it is an element of $\{A | B \succ A \in \Delta\}$.

□

Thus we have completeness. The system can be shown to be sound in the usual manner.

8 Belief Revision from Conditionals

A belief revision system can be defined from a conditional and a satisfaction relation in a simple manner. We can define $K * A$ to be exactly those sentences B such that the conditional $A > B$ is true in the rich state of belief K.

Definition: 9 (Belief Revision based on $>$ and $\models_T$) *The belief revision system based on a conditional $>$ and a satisfaction relation $\models_T$, and a set of belief states $\mathcal{K} = \mathcal{W}_\mathcal{P}$, is the pair consisting of the set of belief states and the function $*_{>,\models_T}$, from belief states*

to belief states, such that for all $K \in \mathcal{K}$ and sentences A, B:

$$(B \in K *_{>,\models_T} A) \equiv (K \models_T A > B)$$

This definition is sometimes referred to as the Ramsey Test. We now define a belief revision system based on the set of belief states in our model of $\models_T$ and the conditional defined in terms of the material and counterfactual conditional.

Lemma: 5 (>) *The belief revision conditional $>$ was defined in terms of $\succ$ and the material conditional. We note that:*

$$\mathbb{S} \models_T A > B \equiv \text{ if } \mathbb{S} \models_T \neg A \text{ then } \mathbb{S} \models_T A \succ B \text{ else } \mathbb{S} \models_T A \rightarrow B$$

Proof: We expand the definitions of $\models_T$. □

Definition: 10 *The belief revision system $\mathfrak{C}$ is defined to be the pair $\langle \mathcal{W}_\mathcal{P}, *_{>,\models_T} \rangle$.*

Theorem: 1 *The belief revision system $\mathfrak{C}$ obeys the AGM postulates.*

Proof: We consider each of the AGM postulates in turn.

R1 Immediate as both material and counterfactuals right hand side is closed under implication.

R2 Immediate as $\models A \rightarrow A$ and $\models_T A \succ A$.

R3 We have two cases. When $\neg A \in K$ then $K + A = Cn(\bot)$, which is all sentences. When $\neg A \notin K$ we have $K * A = \{B | A \rightarrow B \in K\}$, which is exactly $K + A$.

R4 Again when $\neg A \notin K$ we have $K * A = \{B | A \rightarrow B \in K\}$, which is exactly $K + A$.

R5 We have two cases. Firstly when $\neg A \notin K$ then we have $\not\models \neg A$. We know $K * A = \{B | A \rightarrow B \in K\}$ which is not all sentences, as it does not contain $\neg A$. Secondly when $\neg A \in K$, if $\models \neg A$ then there is no state of affairs that satisfies it, and thus no minimal state. Therefore all sentences are the case. If all sentences were the case, on the chosen state of affairs, it must be because there was no state that modeled A, and thus $\models \neg A$.

R6 Immediate by extensionality of all rules.

R7 We again split on the two cases. Firstly $\neg(A \wedge B) \notin K$ then $K * (A \wedge B) = \{C | A \wedge B \rightarrow C \in K\}$. This is $K + (A \wedge B)$, which is a subset of $(K * A) + B$, as by $(R3)$ $K * A \subset K + A$, and $K + A + B = K + (A \wedge B)$. Secondly $\neg(A \wedge B) \in K$, then the $K * (A \wedge B)$ are the sentences true in the state of smallest state of affairs where $A \wedge B$ are the case. Also $K * A$ is the smallest set of affairs where A is the case. Now either $\neg B \in K * A$, in which case $(K * A) + B$ is all sentences, or $\neg B \notin K * A$, in which case the smallest state of affairs that does not contradict $A \wedge B$ is the smallest set of affairs that does not contradict A, and we have an equality.

R8 Following the previous analysis we have the equality, or if $\neg(A \wedge B) \notin K$, then we the equality by $K + (A \wedge B) = (K * A) + B$.

□

Theorem: 2 *The belief revision system $\mathfrak{C}$ obeys the postulate* **Counter**.

Proof: We note that if $\neg A \in K$ then $A > B$ is defined to be $A \succ B$. □

Theorem: 3 *The postulate* **Counter** *and AGM postulates uniquely specify a revision.*

Proof: We assume that we are revising a belief state K with a sentence A. We have two cases. Either $\neg A \in K$, in which case the new state of belief is exactly $\{B | A \succ B \in K\}$, or $\neg A \notin K$ in which case $K = K + A$. □

Theorem: 4 *The AGM postulates are derivable from* **Change**, *the axioms of* **V**, *and consistency preservation.*

Proof: We consider the other 6 postulates in turn.

- R1: This follows from Conjunction and the replacement of logical equivalents.
- R2: This follows from $A > A$, and $A \rightarrow A$.
- R3: This follows from **Change**, and the fact that if $\neg A \in K$, then $K + A$ is all sentences.
- R4:, This is the forward direction of the first conjunct of **Change**.
- R5: This follows from consistency preservation, as if $\bot \in K$ then $A \in K$, so we use **Counter** to revise, and **Counter** is a consequence of **Change**.

- R6: This follows from the replacement of logical equivalents.

- R7: If $A \wedge B$ is consistent with K then this follows from **Change**. Otherwise, if A is consistent with K, then $K * A = K + A$. Now if B is inconsistent with $K + A$, then $(K * A) + B$ is all sentences, and thus $R7$ is true. If B is consistent with $K+A$, then $(K * A) + B$ is $K + A + B$, which is $K + (A \wedge B)$, but this is $K * (A \wedge B)$, thus $R7$ is established in this case. This leaves the possibility that A is inconsistent with K. In this case we have two sub-cases, either $K * A$ is inconsistent with B, in which case, $(K*A)+B = Cn(\perp)$, establishing $R7$, or $K * A$ is consistent with B. Thus $(K * A) + B$ is $\{C | A \succ B \rightarrow C \in K\}$. We also know that, by the fact that $A \wedge B$ is inconsistent with K if A is, that $K * (A \wedge B)$ is $\{C | A \wedge B \succ C \in K\}$. Thus we need to show that $A \wedge B \succ C \in K$ and $\neg(A \succ \neg B) \in K$ implies that $A \succ B \rightarrow C \in K$. But this is derivable from Or, Cautious monotonicity, and Rationality.

- R8: We have two cases, either $\neg A$ is in K, in which case, $K * A = \{C | A \succ C \in K\}$, thus $(K * A) + B = \{C | A \succ B \rightarrow C \in K\}$. But as we know $\neg B \in K * A$, we need to show that $A \succ (B \rightarrow C) \wedge \neg(A \succ \neg B) \rightarrow (A \wedge B \succ C)$. But by rationality we have, $A \succ D \wedge \neg(A \succ \neg B) \rightarrow (A \wedge B \succ D)$, so choosing $B \rightarrow C$ for D, and using Conjunction and replacement interderivables we have the required inclusion. The other case is when $\neg A \notin K$, in which case both sides are equivalent to $K + A + B$.

Theorem: 5 *Let $L_{\mathcal{P}}^{classical}$ be a language, then every set of finite sequences of AGM revisions H of a given belief state $K \subset L_{\mathcal{P}}^{classical}$ is captured by a model of $\mathfrak{C}$, in the sense that there is a rich state of belief $S(H)$ such that,*

$$\text{If } B \in (\ldots((((K *_1 A_1) *_2 \ldots) *_n A_n \in H \text{ then}$$
$$S(H) \models_T (A_1 > (A_2 > \ldots > (A_n > B))))) \ldots)$$

Proof: We construct a rich state of belief that models the set of sequences of AGM revisions H. We do this by using the Grove models for each of the revision operators.

We construct the rich state of belief by induction on the depth of nesting of revisions that are not conjunctions in H.

If there are no revisions that are not modeled by conjoining the sentences, then we model H by the set of worlds that model K, and an arbitrary order on rich state of belief.

If the nesting in H of revisions that are not modeled by conjoining the sentences is n, then construct a rich state of belief from the Grove model, in the following ways. We take the minimal worlds in the Grove model of the first revision as our set of worlds. We replace his total-pre-order on worlds, by the rich states of belief, with those sets of worlds, that correspond to the sub-sequences of revisions that have a nesting of $n-1$ revisions that are not modeled by conjoining the sentences.

As Grove's model is a model of any set of sequences of revisions, where each sequence contains at most one revision $K' * A$, such that $\neg A \in K'$, and that revision, if it exists, is the last revision in the sequence, we know that this rich state of correctly captures the first counterfactual revision. However, as the result of revising the rich state of belief, is another rich state of belief, that correctly captures the next sequence of revisions of this type, we correctly model all finite sets of sequences of revisions□.

It is worthwhile noting that if we did not specify that each revision operator was different, the set of revisions would be more constrained, as belief states that agreed on all conditional free sentences would necessarily revise in the same manner due the functional nature of $*$. As we have noted before this does not apply in our system has the conditional sentences allow multiple belief states that agree on all objective, or conditional free, beliefs.

9 Is $\succ$ a counterfactual?

We have called $\succ$ a counterfactual up until now, and indeed we have axiomatized it as a counterfactual following Lewis. However, it does not exactly correspond to the counterfactual sentences that we would normally affirm. There are many examples of times when we will affirm a counterfactual, but upon learning the antecedent deny the consequent. A classic example is the counterfactual,

> If Hitler had tried to invade Britain he would have won World War II.

If someone who believed this was to learn that Hitler did indeed try to invade Britain, he would not then change his mind on who the victor of World War II was.

The difference here is that when judging counterfactuals we compare worlds based on similarity, after having

chosen to modify certain aspects. Thus when we order worlds in counterfactual similarity orderings we order what worlds we imagine were possible, if things had been different. In the similarity ordering we used for belief revision, it seems that we compare worlds, without discounting any of the differences between that world and reality. Thus we order the worlds that we consider possible without assuming that things are different.

It is superficially surprising that the same rules apply to how we change our beliefs, when we are forced to assimilate new information, and when we imagine how things could be different. However, as both are naturally modeled by orders on possible states of affairs, it is not strange that the differences can be concentrated into which particular order is considered appropriate. This is essentially the same approach as Stalnaker[Stalnaker, 1968, Stalnaker, 1985] has used in differentiating between indicative and subjunctive conditionals.

10 Tabula Rasa

Giving the set of valid sentences that a reasoner will always believe is not the most useful set of sentences to describe. A much more useful set would be the set of sentences a reasoner should believe when told nothing. We could then calculate the sentences a reasoner should believe after being told a sequence of sentences, by applying the rule **Change**.

We now describe one possibility for a reasoner that has been given no information, but which is willing to take information without any biases.

We suggest that when we are told a sentence A with a depth of left nesting of conditionals n we should hold onto all sentences with a higher left nesting of conditionals. It is worth noting that in our system, no consistent set of sentences, each of which has a particular left nesting of conditionals entails any non-valid sentence with an essentially higher left nesting of conditionals. Thus we can treat each set of sentences of a particular nesting independently. We first define left nesting, and then prove this fact.

Definition: 11 *The* nestings *of a sentence containing conditionals are finite binary trees. The nesting of a letter is a leaf, the nestings of a conjunction $A \wedge B$ is the union of the nestings of A and B, the nesting of $\neg A$ is the nesting of A, while the nestings of $A \succ B$ are the trees whose left subtree is a nesting of A and whose right subtree is a nesting of B.*

The left nesting *of A is the depth of the longest left branching path in the nestings of A.*

Lemma: 6 *If we have a set of sentences Γ, all of whose elements have left nestings of conditionals less than n, then $\not\vdash_T \Gamma \rightarrow A$, if A is a sentence whose left nesting of conditionals is greater than n, and A is not equivalent to a sentence whose left nesting of conditionals is less than or equal to n , and $\not\vdash_T A$.*

Proof:

We consider a model of the sentences of Γ. When we write out the truth conditions of Γ, from a rich state of belief S, we say that a occurrence of a rich state of belief S' is nested once, if it the state of belief selected by a $\downarrow A$, from the state S, and we say that an occurrence of a rich state of belief S'' is nested $n+1$ times, if it the state of belief selected by a $\downarrow A$, from the state S''' that is nested n times.

Clearly, none of the sentences of Γ depend on what rich states of the world are ordered under n nestings. But, by assumption, A is not equivalent to a sentence whose right nesting of conditionals is less than or equal to n, therefore, it depends on some feature of the model, that is nested at least n deep. Thus we can modify any model of Γ, so that it is not a model of A, as required.□

We now define the formulas that we should believe in out initial state of belief.

Definition: 12 *For a given sentence $A_n = A_n^{l1} \succ (\ldots A_n^{li} \succ (\ldots (A_n^{lm} \succ A_n^r)\ldots)\ldots)$, of left nesting n, where A_n^r has no conditional in the scope of a negation, we write $\mathcal{A}_{n-1}$ for the set of A_n^{li}'s, of right nesting $n-1$. $\mathcal{A}_{n-1}$ can be ordered with the sub-formulas that occur further to the right being higher. We write $\mathcal{A}_{n-1}^*$ for the largest upwards closed consistent subset of $\mathcal{A}_{n-1}$. We write $\mathcal{A}_{n-1}^+$, for the subset of $\mathcal{A}_{n-1}^*$, which has no conditional in the scope of an odd number of negations.*

We take as formulas of our initial state,

$$\{A_n | \mathcal{A}_{n-1}^+ \vdash_T A_n^r\}$$

$$\{\neg(A_n) | \mathcal{A}_{n-1}^+ \not\vdash_T A_n^r\}$$

That is the formulas, whose largest upwards closed consistent positive subset of $\mathcal{A}_{n-1}$ entail their right hand side.

We now show that all sentences are decided by these sentences, and that this is a consistent state of belief.

Theorem: 6 *Every sentence A is decided by the sentences of our initial state.*

Proof:

We prove this by structural induction. Let the set of sentences of our initial state be Θ

- $A = p$. Take $n = 0$, and $m = 0$ and $A_0^r = p$, then $\not\vdash_T A_0^r$, and thus $\Theta \vdash_T \neg p$.
- $A = B \wedge C$. By the induction hypothesis B and C are decided, thus $B \wedge C$ is decided.
- $A = B_1 \succ (\ldots \succ (B_n \succ C)\ldots)$. Let the nesting depth of C be f. We reduce the value of f by the following construction. If the top-level connective of C is $\wedge$, $C = C_1 \wedge C_2$, then the truth of A is reducible to the truth of $B_1 \succ (\ldots \succ (B_n \succ C_1)\ldots)$ and $B_1 \succ (\ldots \succ (B_n \succ C_2)\ldots)$. Thus we may assume that the top level connective is not $\wedge$. If the top level connective is $\neg$, $C = \neg C_1$, then the truth of A is reducible to the falsity of $B_1 \succ (\ldots \succ (B_n \succ C_1)\ldots)$.

  Thus the top level connective may be taken to be $\succ$, $C = C_1 \succ C_2$, and thus is of the form $B_1 \succ (\ldots \succ (B_n \succ (B_{n+1} \succ C_2)\ldots)$. Thus by this method we may reduce the complexity of C.

  Finally the induction base, $A = B_1 \succ (\ldots \succ (B_n \succ p)\ldots)$, is established as, p is either derivable from the largest upwards closed consistent positive subset of the B's or it is not.

□

Theorem: 7 *The sentences of the initial state, Θ, are consistent.*

Proof: We prove this noticing that any finite subset of the sentences of our initial state can be rewritten into m sentences of the form, $B_1 \succ (\ldots B_n \succ C)$ where each of the m sequences of B's elements are logically distinct. However, the truth conditions of these sentences are all independent. Thus it suffices to show that each sentence is consistent. We note that C will be equivalent to $\bot$ if and only if $\neg B_n$ is valid, by construction. Thus, each of these sentences are consistent, as any sentence of this form is consistent, just so long as $C \equiv \bot$ only when $\vdash_T \neg B_n$. □

Thus we have generated a consistent set of sentences, that uniquely picks out a particular rich state of belief. This state of belief has the property that it revises its belief in a simple bias free way. It captures the natural idea that we only believe the propositions for which we have evidence. Here evidence can be thought of as being told a set of consistent sentences that entail the proposition, or being told defaults that lead us to revise our beliefs so that we accept the proposition.

This model has the natural property that we never believe a conditional sentence with left nesting n, unless it is valid, or we have been told sentences with left nesting greater or equal to n. Thus we never believe a general principle for revising belief, that is one that is not the result of a more general default, unless it is valid, or we have been told it explicitly.

This system of belief revision has the property that is is rational in the sense of Kraus et al. It could be modified so that it was preferential for objective propositions, however it would still be rational for conditionals, as every belief state decides all conditionals, and thus it is never the case that $(A \succ B)$, $\neg(A \succ \neg(C \succ D))$, and $\neg(A \succ (C \succ D))$. Thus, when $(A \succ B)$, and $\neg(A \succ \neg(C \succ D))$, we have $(A \succ (C \succ D))$, and by cumulativity we can derive, $(A \wedge (C \succ D)) \succ B$.

11 Conclusion

We have developed a model of belief revision that models the AGM postulates. We propose a new postulate that completes the AGM postulates, in that it uniquely specifies a revision. Our model of revision captures all finite sequences of AGM revisions. This should be compared with previous models that only captured one revision. Our model is based on the idea that belief revision can be split into two parts, one that deals with what to do when we are told something consistent with out beliefs, and the other that deals with inconsistencies. We propose the use of counterfactual semantics when we are told an inconsistency. We modify Lewis's semantics for counterfactuals by allowing many possible similarity measures based around the same world, and removing the assumption that counterfactual conditionals has the truth value of the material conditional, when their antecedent is true. Our semantics are based around states of belief, rather than possible worlds, We note that this change in the semantics does not substantially change the axiomatization of Lewis system. We show that our system can represent all sequences of AGM style revisions, and show that it exactly captures the AGM postulates.

We then propose one possible state of belief as the reasonable starting point for a reasoner. This state of belief has the property that the reasoner updates their beliefs by conjoining consistent information, by applying the rules that they have been previously told when they are told inconsistent information, and by believing only what they have been told when they

have been given no advice.

This initial state of belief, and the model of belief revision we give fully specify how a reasoner should change their beliefs over time. It specifies a reasoner that can be told arbitrarily nested conditionals, and can thus model defaults about defaults, in contrast to other non-monotonic and conditional systems where defaults only apply to objective sentences.

Acknowledgments

This research has been partly supported by DARPA contract no. ONR 621915

References

[Adams, 1975] Adams, E. (1975). *The Logic of Conditionals*. D. Reidel, Dordrecht, Netherlands.

[Alchourrón et al., 1985] Alchourrón, C. E., Gärdenfors, P., and Makinson, D. (1985). On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510–530.

[Burgess, 1981] Burgess, J. (1981). Quick completeness proofs for some logics of conditionals. *Notre Dame Journal of Formal Logic*, 22(1):76–84.

[Darwiche and Pearl, 1994] Darwiche, A. and Pearl, J. (1994). On the logic of iterated belief revision. In Fagin, R., editor, *Proceedings of the fifth Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 5–23. Morgan Kaufmann.

[Freund and Lehmann, 1994] Freund, M. and Lehmann, D. (1994). Belief revision and rational inference. Technical Report TR 94-16, The Leibniz center for research in computer science.

[Gärdenfors, 1978] Gärdenfors, P. (1978). Conditionals and changes of belief. *Acta Philos. Fennica*, 30:381–404.

[Gärdenfors, 1986] Gärdenfors, P. (1986). Belief revision and the Ramsey test for conditionals. *Philosophical Review*, 91:81–93.

[Gärdenfors, 1988] Gärdenfors, P. (1988). *Knowledge in Flux*. Cambridge University Press, Cambridge, UK.

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

[Kraus et al., 1990] Kraus, S., Lehmann, D., and Magidor, M. (1990). Nonmonotonic Reasoning, Preferential Models and Cumulative Logics. *Artificial Intelligence*, 44(1):167–207.

[Lehmann, 1995] Lehmann, D. (1995). Belief revision, revised. In Mellish, C., editor, *Proceedings of IJCAI 95*, pages 1534–1540. Morgan Kaufmann.

[Lewis, 1976] Lewis, D. K. (1976). Probabilities of conditionals and conditional probabilities. *Philosophical Review*, 85(1):297–315.

[Lewis, 1991] Lewis, D. K. (1991). *Conditionals*, chapter Probabilities of Conditionals and Conditional Probabilities, pages 76–110. Oxford University Press, New York.

[Plotkin, 1975] Plotkin, G. (1975). Call-by-name, call-by-value and the lambda calculus. *Theoretical Computer Science*, 1:125–159.

[Scott, 1972] Scott, D. (1972). Continuous lattices. In *Proc. 1971 Dalhousie Conference: LNM 274*, pages 97–136. Springer-Verlag.

[Stalnaker, 1968] Stalnaker, R. (1968). *A Theory of Conditionals*, volume 2 of *American Philosophical Quarterly Monograph Series (Nicholas Rescher, ed.*, pages 98–112. Blackwell, Oxford.

[Stalnaker, 1985] Stalnaker, R. (1985). *Inquiry*. MIT Press, Cambridge, MA.

[von Rimscha, 1980] von Rimscha, M. (1980). Mengentheoretische modelle des λk kalkuls. *Arch Math Logik Grundlag*, 20(1-2):65–73.

Preference Logic

Preferential multi-agent nonmonotonic logics: Preliminary report

Ana Maria Monteiro
Institute of Computing
State University of Campinas, Brazil
anammont@dcc.unicamp.br

Jacques Wainer
Institute of Computing
State University of Campinas, Brazil
wainer@dcc.unicamp.br

Abstract

This paper presents a way of constructing a multi-agent nonmonotonic logic from any propositional preference logic (the base logic). The multi-agent logic will "correctly" extend the base logic for each agent, that is, each agent will reason in the base logic, and the reasoning of each agent is independent of each other. If the base logic is propositional circumscription then the extended logic will also correctly deal with default formulas that mix the knowledge of more than one agent. Some of the properties of the extended logics and an example of nonmonotonic multi-agent reasoning are presented.

1 Introduction

The term agent is frequently used in the area of knowledge representation and reasoning. Several different systems have been proposed to model an agent reasoning about its environment, an environment that in many applications includes other agents. Thus, some logics have been proposed to deal with the knowledge of more than one agent [HM92, FHV91], but these logics are monotonic and hence have a very limited capacity of modeling interesting behaviors and communications among the agents.

There has been some multi-agent nonmonotonic logics developed in the last years based on autoepistemic logics [Mor90, Lak93, HM93, PJ95]. This paper we will develop a preference-based nonmonotonic logic for reasoning about the beliefs of many agents.

More specifically, this work proposes a way of defining a multi-agent preferential logic based on a propositional preference relation in such a way that whatever is captured by the propositional preference relation is carried over "correctly" to the multi-agent logic. Conceptually, if $\mathcal{L}$ is a propositional logic, one can define a preferential logic $\mathcal{L}_{\leq}$ based on a preference relation $\leq$ on the models of $\mathcal{L}$. On the other hand one can define a multi-agent logic $\mathcal{L}^*$ based on $\mathcal{L}$, which includes modal operators to represent the beliefs of n agents. This paper proposes a way of defining a preference relation $\preceq$ among models of $\mathcal{L}^*$ in order to define the multi-agent preference logic $\mathcal{L}^*_{\preceq}$.

The next section describes preference logics and the semantics of the $\mathcal{L}^*$ logic. Section 3 describes how to construct the $\preceq$ preference relation based on the $\leq$ relation. The other section describes some of the properties of the $\mathcal{L}^*_{\preceq}$ logic. Section 5 describe some examples of using the logic.

2 Background

2.1 Preference Logics

This work is based on the model-theoretic approach to constructing nonmonotonic logics proposed by Shoham [Sho87]. The main idea behind preferential logics is that the meaning of a formula is not given by the set of all of its models, as in classical logics, but by some preferred subset of those models, called *minimal models*. Many different preference criteria can be adopted according to one's needs, resulting in different nonmonotonic logic.

Formally, let $\mathcal{L}$ be a standard (monotonic) propositional logic and $\leq$ a pre-order on interpretations for $\mathcal{L}$. If M_1 and M_2 are two interpretations for $\mathcal{L}$, then $M_1 \leq M_2$ means that the interpretation M_1 is as preferred as the interpretation M_2. $\mathcal{L}$ and $\leq$ define a preference logic $\mathcal{L}_{\leq}$. The language of $\mathcal{L}_{\leq}$ is the same of that of $\mathcal{L}$ and its semantics is defined as follows.

Definition 1 *Given an interpretation M of $\mathcal{L}$ and α a sentence of $\mathcal{L}$, then M preferentially satisfies α (written $M \models_{\leq} \alpha$) if $M \models \alpha$, and if there is no other interpretation M' such that $M' < M$ and $M' \models \alpha$. In this case we say that M is a preferred model of α.*

Definition 2 *α is preferentially satisfiable if there exists an M such that $M \models_{\leq} \alpha$.*

Definition 3 *α preferentially entails β (written $\alpha \models_{\leq} \beta$) if for any M, if $M \models_{\leq} \alpha$ then $M \models \beta$, or equivalently, if the models of β are a superset of the preferred models of α.*

This framework is general enough to capture the details of many nonmonotonic reasoning systems. Among the minimal model approaches, we can mention circumscription [McC80, McC86], and many proposals for logics that reason about time and change [Sho88, Bak91, Sha95].

In this paper we will be restricted to propositional logics, and we will develop examples of default reasoning (as opposed to, say, temporal reasoning). There is a trivial simplification of circumscription that allows one to represent and reasoning about defaults in a propositional language. We call it propositional circumscription. Default formulas of the kind α then normally β are represented as:

$$\alpha \wedge \neg abn_1 \rightarrow \beta$$

The preference criteria is the one that tries to falsify abn_1. That is: $M_1 < M_2$ iff

- for all propositional symbols p in the language other than abn_1 $M_1 \models p$ iff $M_2 \models p$.
- $M_1 \models \neg abn_1$ and $M_2 \models abn_1$

All the standard circumscription variations: priority among abnormal predicates, formula circumscription and allowing predicates to vary, can be implemented in propositional circumscription.

2.2 The $\mathcal{L}^*$ logic

In this section we describe how to extend a monotonic base logic $\mathcal{L}$ to a monotonic multi-agent logic $\mathcal{L}^*$. The language of $\mathcal{L}^*$ is the same of $\mathcal{L}$, augmented with a set of new modal operators $\{B_1, \ldots, B_n\}$ to represent the beliefs of each of the agents 1 ..., n. That is, if α is a well-formed formula of the language of $\mathcal{L}$, then it is also a well-formed formula of $\mathcal{L}^*$, and if α is a well-formed formula of $\mathcal{L}^*$, so is $B_i\alpha$ for $i \in \{1, \ldots, n\}$.

To model the states of belief of the agents we use the belief structures introduced by Fagin, Halpern and Vardi [FHV91]. The idea of possible worlds is employed here in a lightly different way of that of traditional Kripke structures. The worlds are defined inductively:

Definition 4 [1] *A 0th-order belief assignment, f_0 is a truth assignment to the primitive propositions. We call $\langle f_0 \rangle$ a 1-ary world, (since its "length" is 1).*

Assume inductively that k-ary worlds (or k-worlds, for short) have been defined. Let W_k be the set of all k-worlds. A **k-th-order belief assignment** *is a function $f_k : A \rightarrow 2^{W_k}$, where A is the set of agents. A* **(k+1)-sequence** *of belief assignment is a sequence $\langle f_0, \ldots, f_k \rangle$, where f_i is a i-th-order belief assignment.*

A **(k+1)-world** *is a (k+1)-sequence of belief assignments that satisfy the following restrictions for each agent i:*

1. *$f_k(i)$ is nonempty if $k \geq 1$.*
2. *If $\langle g_0, \ldots, g_{k-1} \rangle \in f_k(i)$ and $k > 1$, then $g_{k-1}(i) = f_{k-1}(i)$.*
3. *$\langle g_0, \ldots, g_{k-2} \rangle \in f_{k-1}(i)$ iff there is a (k-1)st-order belief assignment g_{k-1} such that $\langle g_0, \ldots, g_{k-1} \rangle \in f_k(i)$, if $k > 1$.*

Intuitively, a 1-ary world is a description of reality and a belief assignment f_k associates with each agent a set of "possible k-worlds"; that is, the worlds in $f_k(i)$ are the k-worlds that agent i thinks are possible descriptions of the reality (as described by k-worlds).

The belief structures are defined based on these k-worlds.

Definition 5 *An infinite sequence $\langle f_0, f_1, \ldots \rangle$ is called a belief structure if each prefix $\langle f_0, \ldots, f_k \rangle$ is a k-world, for each k.*

Thus a k-world describes beliefs of depth $k-1$, where the depth of a formula is roughly the number of nested belief operators in the formula. A belief structure describes beliefs of arbitrary depth.

For a formula α, the worlds that have the information needed to evaluate it are the k-worlds such that $k \geq \text{depth}(\alpha)$.

Definition 6 *A $(k+1)$-world $\langle f_0, \ldots, f_k \rangle$ satisfies a formula α, written $\langle f_0, \ldots, f_k \rangle \models \alpha$ if $k \geq depth(\alpha)$ and:*

- *$\langle f_0, \ldots, f_k \rangle \models \alpha$ iff $f_0 \models \alpha$ for α a propositional formula.*
- *$\langle f_0, \ldots, f_k \rangle \models \neg\alpha$ iff $\langle f_0, \ldots, f_k \rangle \not\models \alpha$;*
- *$\langle f_0, \ldots, f_k \rangle \models \alpha \wedge \beta$ iff $\langle f_0, \ldots, f_k \rangle \models \alpha$ and $\langle f_0, \ldots, f_k \rangle \models \beta$;*
- *$\langle f_0, \ldots, f_k \rangle \models B_i\alpha$ iff for each $\langle g_0, \ldots, g_{k-1} \rangle \in f_k(i)$, $\langle g_0, \ldots, g_{k-1} \rangle \models \alpha$*

But it suffices to consider the $depth(\alpha)$-worlds.

Proposition 1 *Assume that $depth(\alpha) = r$ and $k \geq r$. Then, $\langle f_0, \ldots, f_k \rangle \models \alpha$ iff $\langle f_0, \ldots, f_r \rangle \models \alpha$.*

In view of the last result the notion of satisfiability of a formula is captured by the following definition:

[1] The definition of belief-world is similar to the one presented in [FHV91] with the exception of restriction 1. This change yields a logic in which each B_i follows the axioms of KD45 (for belief) instead of [FHV91]'s S5 axioms (for knowledge).

Definition 7 *We say that the belief structure* $f = \langle f_0, f_1, \ldots \rangle$ *satisfies* α, *written* $f \models \alpha$, *if* $\langle f_0, \ldots, f_r \rangle \models \alpha$, *where* $r = depth(\alpha)$.

Definition 8 *We say that a formula* α *is valid, written* $\models \alpha$ *if* $f \models \alpha$ *for each belief structure* f.

The logic $\mathcal{L}^*$ just defined has the following properties.

Proposition 2

- *All substitution instances of the tautologies are valid.*
- $\models B_i\alpha \rightarrow \neg B_i \neg \alpha$.
- $\models B_i\alpha \rightarrow B_i B_i \alpha$.
- $\models \neg B_i\alpha \rightarrow B_i \neg B_i \alpha$.
- $\models B_i\alpha \wedge B_i(\alpha \rightarrow \beta) \rightarrow B_i\beta$.

Thus, each of B_i satisfy the requirements of the logic KD45, which are the most accepted logic for belief.

3 The logic $\mathcal{L}^*_{\preceq}$

3.1 The $\preceq$ preference relation

Given the monotonic multi-agent logic $\mathcal{L}^*$, if we define a preference relation $\preceq$ on the set of belief structures of $\mathcal{L}^*$, then we would have a multi-agent preference logic. This section describes how to define a preference relation $\preceq$ based on a preference relation $\leq$ defined on the set of propositional interpretation of $\mathcal{L}$.

We would like to use the inductive definition of k-worlds to define a preference relation among them. Clearly, $\leq$ is exactly the preference relation defined for 1-worlds (which are just a propositional assignment). The difficulty is that a 2-world attributes to each agent, a *set of* 1-worlds. Thus we have to define a preference relation among sets of 1-worlds, given that there is a preference $\leq$ among 1-worlds. Or in general, given a pre-order $\leq$ on a set A, we must define a pre-order $\sqsubseteq$ on the set 2^A.

We will follow idea of elementary improvement discussed in [Wai93]. The set A_1 is an elementary improvement ($\sqsubset_e$) over the set A_2, if they agree in all worlds, except for a worlds in A_1 which is better (under the $\leq$ order) than its correspondent in A_2. More formally, $A_1 \sqsubset_e A_2$, if $A_1 = A \cup \{a_1\}$ and $A_2 = A \cup \{a_2\}$ and $a_1 \leq a_2$; or $A_2 = A_1 \cup \{a_2\}$ and there is a a_1 in A_1 such that $a_1 \leq a_2$[2].

We are interested in the transitive, reflexive closure of the $\sqsubset_e$ relation, which has a rather simple formulation:

[2]This last case corresponds to the situation where the world that "got better" in the passage from A_2 to A_1 "landed" on a world that was "already there".

Definition 9 *Let* $\leq$ *a pre-order defined on a set* A. *We define the relation* $\sqsubseteq$ *on the set* 2^A *as:* $A_1 \sqsubseteq A_2$ *iff for all* $a \in A_1$ *there is* $b \in A_2$ *such that* $a \leq b$ *and for all* $b \in A_2$ *there is* $a \in A_1$ *such that* $a \leq b$.

It can be shown that $\sqsubseteq$ is a pre-order in 2^A.

Now we can inductively define the pre-order $\preceq_k$ among k-worlds.

Definition 10 *Let* $\leq$ *be a pre-order defined on the set of propositional interpretations, then one can inductively define* $\preceq_k$ *as a pre-order on the set of k-worlds as:*

- $\preceq_1 = \leq$.
- *if* $\preceq_k$ *has been defined on the set of k-worlds, then* $\preceq_{k+1}$ *is defined on the set of* $k+1$*-worlds as:*

  $\langle f_0, \ldots, f_k \rangle \preceq_{k+1} \langle f'_0, \ldots, f'_k \rangle$ *iff*
  $\langle f_0, \ldots, f_{k-1} \rangle \preceq_k \langle f'_0, \ldots, f'_{k-1} \rangle$ *and for each* i *such that* $1 \leq i \leq n$, $f_k(i) \sqsubseteq_k f'_k(i)$, *where* $\sqsubseteq_k$ *is defined from* $\preceq_k$, *as in definition 9.*

And finally we can define the preference relation $\preceq$ among belief structures:

Definition 11 *Let* $\leq$ *a pre-order on a set of propositional interpretations. If* f *and* f' *are belief structures, then* $f \preceq f'$ *iff for each prefix* $\langle f_0, \ldots, f_{k-1} \rangle$ *of* f *and for each prefix* $\langle f'_0, \ldots, f'_{k-1} \rangle$ *of* f', $\langle f_0, \ldots, f_{k-1} \rangle \preceq_k \langle f'_0, \ldots, f'_{k-1} \rangle$.

$\mathcal{L}^*$ together with the pre-order $\preceq$ defines a multi-agent preferential logic $\mathcal{L}^*_{\preceq}$ that allows one to model the beliefs of agents that can reason non-monotonically.

4 Properties of the logic $\mathcal{L}^*_{\preceq}$

The logic $\mathcal{L}^*_{\preceq}$ defined above have some interesting properties. The proofs of the theorems in this section and the next can be found at [Mon96].

The first important and potentially controversial property of the logic is that all agents have the same reasoning power. In particular, when reasoning about formulas that do not contain modal operators, all agents use the preference logic defined by $\leq$. Section 4.2 discusses the implications of this theorem, and shows that the assumption that all agents reason with the same preference does not limit the applicability of the logic $\mathcal{L}^*_{\preceq}$ in a multi-agent situation.

Theorem 1 *If* α *and* β *are propositional formulas then* $\alpha \models_{\leq} \beta$ *iff* $B_i\alpha \models_{\preceq} B_i\beta$.

The next theorem states that it is common knowledge that all agents reason (about the world) based on the same propositional preference:

Theorem 2 *If* $\alpha \models_{\preceq} \beta$ *then* $B_{i_1} \ldots B_{i_r} \alpha \models_{\preceq} B_{i_1} \ldots B_{i_r} \beta$, *whenever* $r > 0$.

Another result is related to the independence of the reasoning performed within different belief contexts.

Theorem 3 *If* $\alpha \models_{\preceq} \beta$ *and* $\gamma \models_{\preceq} \delta$ *then* $B_i \alpha \wedge B_j \gamma \models_{\preceq} B_i \beta \wedge B_j \delta$ *if* $i \neq j$.

This states that inferences that could be performed independently in $\mathcal{L}_{\preceq}$ can still be performed if they are in different belief contexts. In particular, it could be the case that in $\mathcal{L}_{\preceq}$, α and γ could interfere with each other, for example $\alpha \wedge \gamma \models_{\preceq} \neg\beta$, but that interference is prevented by the different belief contexts.

Theorem 3 can be extended to arbitrary belief contexts. To state the extended theorem we need to define a syntactic operation on sequences of B_i operators: $\overline{Q}$ is a operation that substitutes in Q all subsequences of adjacent B_i by a single B_i. For example:

$$\overline{B_3 B_3 B_7 B_7 B_7 B_2 B_2 B_7} = B_3 B_7 B_2 B_7$$

Theorem 4 *If* $\alpha \models_{\preceq} \beta$ *and* $\gamma \models_{\preceq} \delta$ *then* $Q_a \alpha \wedge Q_b \gamma \models_{\preceq} Q_a \beta \wedge Q_b \delta$ *where* Q_a *and* Q_b *are any sequence of* B_i *operators, provided that* $\overline{Q_a} \neq \overline{Q_b}$.

The intuition for the restriction on Q_a and Q_b is that if $\overline{Q_a} = \overline{Q_b}$ then $Q_a \alpha$ is equivalent to $Q_b \alpha$.

4.1 Mixed defaults

The theorems above state generic properties of the $\mathcal{L}^*_{\preceq}$ logic, independently of the $\preceq$ base preference relation. We will now show that if the base logic is propositional circumscription then default rules that combine subformulas with and without the modal operators are correctly dealt with. We call such default rules as **mixed defaults**.

If $\preceq_a$ is the preference that minimizes the propositional symbol abn_1, and if $\preceq_a$ is the corresponding preference for the $\mathcal{L}^*$ models, then the following results are true:

- $B_i \alpha \wedge (B_i \alpha \wedge \neg abn_1 \rightarrow \beta) \models_{\preceq_a} \beta$.
- $\alpha \wedge (\alpha \wedge \neg abn_1 \rightarrow B_i \beta) \models_{\preceq_a} B_i \beta$.
- $\alpha \wedge (\alpha \wedge \neg abn_1 \rightarrow B_i \beta) \wedge B_i(\neg\beta) \not\models_{\preceq_a} B_i \beta$

In fact, results above hold when B_i is substituted by any sequence $B_{i_1} B_{i_2} \ldots B_{i_r}$ for $r > 0$. The first two examples show that if the modal sub-formula is either the antecedent or the consequent of a default rule, and the antecedent is true, then the default consequent would be inferred. The last example shows that the logic when determining if a default consequence can be consistently asserted, will take into consideration the logic of the B_i operators.

4.2 Discussion

We must address now the question brought up by theorem 1 which states that all agents reason propositionally using the same preference $\preceq$. This may seems too strict, and one could think that this assumption would limit the usefulness of the logic for multi-agent applications. The underlying question is: if all the agents reason with the same preference then it is not the case that they are all the "same" agent.

First we would like to point out that the use of the term preference in this paper refers to a pre-order relation among models, and not a relation among propositions. Thus the logic presented here does not model statements like "Agent 1 prefers to have a toothache than to go to the dentist," which one would certainly not want to be the same for all agents. The preference relation in this paper is the implementation of a form of nonmonotonic reasoning. It is not a form of representing this preference aspect of the mental model of agents ([WD91, Wai94] are attempts to represent this mental aspect).

As for the fact that all the agents reason based on the same preference, one should notice that either there is a single preference relation for all the agents, or there is an infinite number of them. It is not just that agent 1 could use a different preference than agent 2, but also that agent 1 could attribute to agent 2 a different preference, which could be also different than the preference agent 1 thinks agent 2 thinks agent 1 uses, and so on. In other words, one would need a preference to reason within the context of a single B_1, possibly another for reasoning within the context of $B_1 B_2$, possibly another to reason within the context of $B_1 B_2 B_1$, and so on.

There are applications of preference logic, specially temporal reasoning [Sho88, Bak91, Sha95], for which it would be acceptable to propose that a single general preference relation that apply to all agents. For temporal reasoning in particular, these logics usually propose a particular preference relation that minimizes "meaningful" predicates or formulas in such a way that reasoning about time and change can be correctly performed. If that particular preference relation is the essence of common-sense reasoning about time and change it is very acceptable to assume that all agents use that same preference to reason about time and change.

Other applications of preference logic, such as taxonomic reasoning and default reasoning, which is the main interest of this work, could possibly benefit from multiple preferences. The crucial difference is that preference logics are used as an *implementation* of default or taxonomic reasoning. One minimizes "meaningless" or arbitrary propositional symbols and thus there is no claim that the preference is unique. For example $\alpha \wedge \neg abn_1 \rightarrow \beta$ implements the default rule

that if α then normally β, if the preference relation minimizes abn_1. But so does $\alpha \wedge \neg abn_2 \rightarrow \beta$, if the preference relation minimizes abn_2. Since the preference relation is in some way arbitrary, it is unreasonable to expect that all agents use the same one.

Surprisingly, the logic $\mathcal{L}^*_{\preceq}$ can deal with a finite number of multiple preference relations. This is done by bringing this finite set of these preferences into the base-logic preference relation $\leq$, provided that the preferences are not contradictory. For example, let us suppose that agent 1 reasons by minimizing the propositional symbol abn_1, and he believes agent 2 reasons by minimizing the preference abn_2 and that he thinks agent 2 thinks he reasons by minimizing abn_3. One can construct a propositional preference relation $\leq$ that simultaneously minimizes abn_1, abn_2, and abn_3. If all formulas within the scope of a single B_1 do not contain either abn_2 or abn_3, then reasoning based on the $\leq$ relation would derive the correct conclusions for the beliefs of agent 1. Similarly if all formulas within the scope of B_1B_2 do not contain either abn_1 or abn_3 then again all reasoning performed in that belief context would be the same as having a particular preference of minimizing abn_2 applied for only that context.

Finally, an important part of default reasoning using preference logic, in particular circumscription, is defining the circumscription policy, that is, which predicates are allow to vary in the minimization process and so on. In a propositional preference logic, circumscription policy is restricted to setting up the priority among the propositional symbols that will be minimized. By "bringing" all these preferences into the $\leq$ relation, one is able to set different priorities for each "particular preference" as to allow the correct chaining of inferences across beliefs contexts, as the example 2 discussed in the section below illustrates, which would not be possible if all preferences were isolated.

5 Examples

Example 1. In order to illustrate an example of multi-agent preferential reasoning, let us assume the following situation. Agent 1 are sitting with a friend (agent 2) in a cafe and waiting for third friend (agent 3). If agent 3 leaves work at 5 (which we will abbreviate as $l5$) she will normally arrive at the cafe at 6 ($a6$), but not so if there is a traffic jam (tj). Furthermore, let assume that all agents know these facts.

The first problem the logic developed herein solves is a representational problem. One can represent, using a preference logic, the default rule that leaving at 5 will normally entail arriving at 6 as:

$$l5 \wedge \neg abn_1 \rightarrow a6 \qquad (1)$$

if the preferential logic is based on the preference relation $\leq_x$ that circumscribes the propositional symbol abn_1. But it is not clear how to represent that agent 1, or agent 2 believes (1) in such a way that the default would work correctly. The main result of this work is that one can develop a multi-agent preference logic based on an already understood propositional preference logic, and one of the results is that each agent in the society will reason (about the real world) using the propositional preference logic. In the logic $\mathcal{L}^*_{\preceq_x}$, where the preference relation $\preceq_x$ is derived from $\leq_x$, one can represent the fact that agent 2 believes (1), as one would expect:

$$B_2(l5 \wedge \neg abn_1 \rightarrow a6)$$

and this representation works correctly, that is

$$B_2(l5) \wedge B_2(l5 \wedge \neg abn_1 \rightarrow a6) \models_{\preceq_x} B_2(a6)$$

Example 1 (cont). Now let us suppose that agent 3 left at 5 and all the agents know it. Agent 2 believes that there is a traffic jam, and he believes that 3 does not know it. On the other hand, agent 1 believes that there is no traffic jam. From this situation one can conclude that

- agent 1 believes agent 3 will arrive at six,
- agent 2 believes agent 3 will not
- agent 2 also believes that agent 3 thinks she will arrive at six .

This reasoning can be captured by the logic $\mathcal{L}^*_{\preceq_x}$.

Let us make the following abbreviation where α is the knowledge that if agent 3 leaves at 5, she would usually arrive by 6, unless there is a traffic jam.

$$\alpha \equiv (l5 \wedge \neg abn_1 \rightarrow a6) \wedge (l5 \wedge tj \rightarrow \neg a6)$$

Then, the situation above is captured by:

$$B_1\begin{pmatrix} l5 \wedge \\ \neg tj \wedge \\ \alpha \end{pmatrix} \wedge B_2\begin{pmatrix} l5 \wedge \\ tj \wedge \\ \alpha \end{pmatrix} \wedge B_2B_3\begin{pmatrix} l5 \wedge \\ \alpha \end{pmatrix}$$
$$\models_{\preceq_x} B_1(a6) \wedge B_2(\neg a6) \wedge B_2B_3(a6)$$

The example above illustrate that defaults that are contained within a B_i operator work as expected.

Example 1 (another solution). The solution above uses the same abnormal symbol for all belief context, or in other words, all agents use the same preference, the one that minimizes abn_1. Let us develop another solution, where each agent uses its own abnormal propositional symbol and its "own" preference. Let us use the following abbreviations:

$$\alpha \equiv (l5 \wedge \neg abn_1 \rightarrow a6) \wedge (l5 \wedge tj \rightarrow \neg a6)$$
$$\alpha' \equiv (l5 \wedge \neg abn_2 \rightarrow a6) \wedge (l5 \wedge tj \rightarrow \neg a6)$$
$$\alpha'' \equiv (l5 \wedge \neg abn_3 \rightarrow a6) \wedge (l5 \wedge tj \rightarrow \neg a6)$$

Now the situation described in the second part of Example 1 is represented as:

$$B_1\begin{pmatrix} l5 \wedge \\ \neg tj \wedge \\ \alpha \end{pmatrix} \wedge B_2\begin{pmatrix} l5 \wedge \\ tj \wedge \\ \alpha' \end{pmatrix} \wedge B_2B_3\begin{pmatrix} l5 \wedge \\ \alpha'' \end{pmatrix} \qquad (2)$$

In order to define a multi-agent preference logic, one need to bring those three preferences (minimizing abn_1 in the belief context B_1, minimizing abn_2 in the context B_2 and abn_3 in the context B_2B_3) into a single propositional preference. This is done by defining the propositional preference $\leq_y$ which minimizes in parallel (that is with the same priority) abn_1, abn_2, and abn_3. That is, $M_1 \leq_y M_2$ iff:

- for all propositional symbols p in the language other than abn_1, abn_2, abn_3, mbox $M_1 \models p$ iff $M_2 \models p$.
- there is no i such that $M_1 \models abn_i$ and $M_2 \models \neg abn_i$

Once $\leq_y$ is defined, the logic $\mathcal{L}^*_{\preceq_y}$ would allow one to derive the correct conclusions from (2), that is:

$$(2) \quad \models_{\preceq_y} \quad B_1(a6) \wedge B_2(\neg a6) \wedge B_2B_3(a6)$$

Example 2. Let us see now that mixed defaults also work as expected. Let us add to the common knowledge the fact that if it is raining (ra) then usually there is a traffic jam, and the fact that if it is raining then agent 2 knows it (because he hears the noise).

We will use the same scheme as the first solution to example 1, that is to use the same abnormal propositional function in all contexts where it is appropriate. Let us make the following abbreviations:

$$\alpha \equiv (l5 \wedge \neg abn_1 \rightarrow a6) \wedge (l5 \wedge tj \rightarrow \neg a6)$$
$$\beta \equiv (ra \wedge \neg abn_2 \rightarrow tj)$$
$$\gamma \equiv (ra \wedge \neg abn_3 \rightarrow B_2 ra)$$

α is the statement that leaving at 5 normally results in arriving at 6, unless there is a traffic jam, β states that if it is raining then normally there is a traffic jam, and γ states that it it is raining then normally agent 2 will know it.

Let us now suppose that agent 1 believes (or sees) that it is raining, then

- he concludes that agent 2 knows it (by γ)
- he concludes that agent 2 believes there is a traffic jam (by β)
- he concludes that agent 2 believes that agent 3 will not arrive at six.

For the reasoning above to work one needs that defaults represented by β and γ to be stronger than the competing default α. But it is important also to notice that these defaults will be operating in different belief contexts: γ is used inside the B_1 operator and β and α are used inside the B_1B_2 sequence of operators.

If one had a preference for each belief context, it is not clear how to set up the priorities among the defaults. But in our approach these preferences are brought together at the $\leq$ preference relation. If one wants γ and β to be stronger than α, one defines a preference relation $\leq_z$ where the propositional symbols abn_2 and abn_3 are minimized with higher priority than abn_1. The logic that extends the relation $\leq_z$ will yield the correct deductions:

$$B_1 ra \wedge B_1(ra \wedge \neg abn_3 \rightarrow B_2 ra) \wedge B_1B_2(l5) \wedge B_1B_2(\beta) \models_{\preceq_z} B_1B_2(\neg a6)$$

This example illustrates a form of chaining across belief contexts: defaults in the B_1's belief space ($ra \wedge (ra \wedge \neg abn_3 \rightarrow B_2 ra)$) trigger defaults in agent 2 belief space ($B_2(l5 \wedge \beta)$). In agent 2 belief space the default that $l5$ normally entail $a6$ is disabled by the higher priority of the default that if ra then $\neg a6$.

6 Conclusions and Future work

As far as the authors know this is the first preference based nonmonotonic logic for multi-agents. The other examples of non-monotonic multi-agent logics [Mor90, Lak93, HM93, PJ95] follow the autoepistemic tradition. One important difference between the preferential and autoepistemic logics is that the former allows one to easily introduce priorities among the defaults, as we did in example 2. It will be an interesting extension of this work to compare a flat version of the preference multi-agent logic (that is, where all minimizations are done in parallel) with some of these autoepistemic logics.

The work presented here is still preliminary. First, we need to discover other properties of the logic $\mathcal{L}^*_{\preceq}$. For example, we cannot yet present any result on how lack of knowledge ($\neg B_i$) interacts with the preference ordering.

It is also important to extend the results for a quantified logic since preference logics have naturally been based on quantified logics.

Acknowledgments

The first author was supported by Universidad Nacional de La Plata and by a scholarship from CNPq, and the second author was partially supported by CNPq grant 521464/93-5. This work is part of the CNPq PROTEM III project LOGIA.

References

[Bak91] A. B. Baker. Nonmonotonic reasoning in the framework of situation calculus. *Artificial Intelligence*, 49:5-23, 1991.

[FHV91] R. Fagin, J. Y. Halpern, and M. Y. Vardi. A model-theoretic analysis of knowledge. *Journal of the A.C.M.*, 32(2):382-428, 1991.

[HM92] J. Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54(3):319–379, 1992.

[HM93] J. Y. Halpern and Y. O. Moses. Reasoning about only knowing with many agents. In *Proc. of the 11th National Conference on Artificial Intelligence (AAAI)*, pages 655–661, 1993.

[Lak93] G. Lakemeyer. All they know: a study in multi-agent autoepestemic reasoning. In *Proc. Thirteenth International Joint Conference on Artificial Intelligence (IJCAI '93)*, 1993. Unpublished manuscript.

[McC80] J. McCarthy. Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[McC86] J. McCarthy. Applications of circumscription to formalizing common-sense reasoning. *Artificial Intelligence*, 28:89–116, 1986.

[Mon96] A. M. Monteiro. Estensões epsitemicas de duas familias de logicas não monotonicas Master thesis *in portuguese*. Instituto de Computação, UNICAMP, Campinas 13083-970, Brazil, 1996

[Mor90] L. Morgenstern. A theory of multiple agent nonmonotonic reasoning. In *Proc of the AAAI-90*, pages 538–544, 1990.

[PJ95] Y. Permpoontanalarp and J. Y. Jiang. Generalized proof-theory for multi-agent autoepistemic reasoning. In *Proceedings of the ICMAS*, pages 304–311, 1995.

[Sha95] M. Shanahan. A circumscriptive calculus of events. *Artificial Intelligence*, 77(2):249–284, 1995.

[Sho87] Y. Shoham. A semantical approach to nonmonotonic logics. In *Proc. of the Tenth International Joint Conference on Artificial Intelligence*, pages 388–392, 1987.

[Sho88] Y. Shoham. *Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence*. MIT Press, Cambridge, MA, 1988.

[Wai93] J. Wainer. Epistemic extension of preference logics. In *Proc. of the 13th International Joint Conference on Artificial Intelligence*, pages 382–387, 1993.

[Wai94] J. Wainer Yet another semantics of goals and goal priorities. In *Proc. of the 11th European Conference on Artificial Intelligence*, pages 269-273, 1994.

[WD91] M. P. Wellman and J. Doyle. Preferential semantics for goals. In *Proc. of the 9th National Conference on Artificial Intelligence*, pages 698-703, 1991.

A representation theorem for preferential logics

Pierre Siegel
LIM - URA CNRS 1787
Centre de Mathématiques et d'Informatique
39, rue Joliot-Curie - 13453 Marseille Cedex 13
téléphone : 91-11-36-01
e-mail : siegel@gyptis.univ-mrs.fr

Lionel Forget
LIM - URA CNRS 1787
Centre de Mathématiques et d'Informatique
39, rue Joliot-Curie - 13453 Marseille Cedex 13
téléphone : 91-11-36-09
e-mail : forget@gyptis.univ-mrs.fr

Abstract

This paper shows how to define nonmonotonic logics from any classical logics $\mathcal{L}$ and any set X of formulas of $\mathcal{L}$. In this context, the nonmonotonic inference relation $\vdash_X$ is defined by $A \vdash_X B$ if every classical theorem of $A \cup B$ which is in X is a theorem of A. The properties of the relation $\vdash_X$ are studied. We show, in particular, that the elementary properties (supraclassicity, or, left logical equivalence, cut, etc.) are verified for any X. Moreover, we prove that cumulativity is verified if the set of formulas of the language, which are not in X, is deductively closed. Then we prove a representation theorem, i.e., in the finite case every preferential nonmonotonic logic is an X-logic. We also study a particular form of the set X for propositional circumscription.

KEYWORDS: *nonmonotonic logic, preferential model approach, representation theorem*

1 INTRODUCTION

Recently, many nonmonotonic logics have been studied. Roughly, it is possible to classify them into two classes. First, the "preferential approach" (more semantical, like circumscription [McCarthy 1980]), the non-monotony property is defined using an order relation between models. For the second approach (more syntactic, like default logics [Reiter 1980]), specific rules are added to describe nonmonotonic phenomena. With this in mind, nonmonotonic logics can be defined by modifying the order relation, the form of defaults, the base language (modal, classical, numerical), etc.

On the other hand, the proof procedures for these different logics use few method. Often, to prove that $A \mid\sim B$, it is necessary to find a finite subset of formulas $F = \{f_1, f_2, \ldots, f_n\}$ such that $\{A \cup F\} \vdash B$, where $\vdash$ denotes a "classical implication". This set F may be called "hypotheses". We encounter approach in preferential logics, defaults logics, ATMS, abduction problems... If the classical implication has a deduction theorem, $A \cup F \vdash B$ is equivalent to $A \wedge \neg F \vdash \neg B$ in the finite case. The proof procedure can then be based on a consequence finding algorithm[Siegel 1987, Inoue 1992], or ATMS.

[Siegel 1981, Bossu and Siegel 1985], on the other hand, define another kind of proof procedure. This proof procedure is defined for the subimplication, which is a logic similar to the GCWA. For this logic, if A is a set of clauses and B is a negative clause, $A \mid\sim B$ if and only if the set of positive clauses entailed by A and B is the same as the set of positive clauses entailed by A alone. As in the previous case, proof procedures are based on consequence finding algorithms.

In this paper we extend this previous definition by defining that from any classical logic and any set X of formulas, the nonmonotonic inference relation $\vdash_X$ is defined by $A \vdash_X B$ if every classical theorem of $A \cup B$ which is in X is a theorem of A. The properties of the relation $\vdash_X$ are studied. We show, in particular, that the elementary properties (supraclassicity, or, left logical equivalence, cut, etc.) are verified for any X. Moreover, we prove that cumulativity is verified if the set of formulas of the language, which are not in X, is deductively closed. Then we prove a representation theorem, i.e., in the finite case every preferential nonmonotonic logic is an X-logic. We also study a particular form of the set X for propositional circumscription. It is clear that the first aim of this paper is not to define an effective logic, but to give a way

to compare existing nonmonotonic logics (in the next step, it will be possible to define new nonmonotonic logics). The second aim of this work is to obtain clear proof procedures for different formalisms.

2 FORMALISM AND PROPERTIES

2.1 DEFINITION

In this paper, $\mathcal{L}$ is a classical logic which inference relation is denoted $\vdash$. $\mathcal{F}$ denotes the set of all formulas of $\mathcal{L}$. If F is a set of formulas of $\mathcal{L}$, $\overline{F}$ denotes the set of theorems of F : $\overline{F} = \{f / F \vdash f\}$.
For us, a classical logic, is a logic, which the language includes connectors $\vee$, $\wedge$, $\neg$ and $\rightarrow$, which :

a) is monotonic ($\overline{A} \subset \overline{A \cup B}$),
b) is idempotent ($\overline{\overline{A}} \subset \overline{A}$),
c) is compact ($F \vdash f$ iff there exists a finite subset F' of F such that $F' \vdash f$),
d) is reflexive ($A \vdash A$),
e) has the deduction theorem :
$F \vdash f \rightarrow g$ ssi $\{F \cup f\} \vdash g$.

For example, boolean algebras, first order logic, normal modal logics are classical logics.
A classical theorem of a set F of formulas is a theorem of F for $\vdash$.

Definition 1 (X-Logic) *Let X a set of formulas of $\mathcal{L}$ (X is not necessary deductively closed). The nonmonotonic inference relation $\vdash_X$ is defined by :*

$$A \vdash_X B \quad iff \quad \overline{A \cup B} \cap X \subset \overline{A} \cap X$$

Therefore $A \vdash_X B$ if every classical theorem of $A \cup B$, which is in X is a theorem of A (by adding B to A the set of classical theorems which are in X does not grow). For the intuition, if a formula is considered as an information, X can be considered as the set of "pertinent" information. Furthermore a set A of informations implies a set B of informations if the addition of B to A does not produce more "pertinent" formulas than with A alone.

As the base logic is monotonic, we have $(\overline{A} \cap X) \subset (\overline{A \cup B} \cap X)$, and therefore it is possible to define $\vdash_X$ by :

$$A \vdash_X B \quad iff \quad \overline{A \cup B} \cap X = \overline{A} \cap X$$

From the definition and the monotonicity of $\vdash$ some obvious properties follow :

1. $\vdash_X$ is not a monotonic inference : If $\vdash A \rightarrow B$ and $B \vdash_X C$ it is possible to have $A \not\vdash_X C$
2. $\vdash_X$ and $\vdash$ are equal on X (if B is in X, then $A \vdash_X B \Leftrightarrow A \vdash B$) and therefore $\vdash_X$ is monotonic on the set X.
3. $\vdash_X \equiv \vdash$ if X is the set of all formulas of $\mathcal{L}$.

Example : If L is a finite propositional logic :

- It is possible to represent the Closed World Assumption (CWA), by choosing X as the set of propositions (positive literrals). The set X can also be the set of positive clauses, or of positive formulas (formulas which contain only connectors $\wedge$ and $\vee$).
- In particular cases of circumscription, the Generalised Closed World Assumption (GCWA) and the sub-implication [Bossu and Siegel 1985] (where all the predicates are circumscribed), X can be the set of conjunction of positive clauses. It can also be the set of positive formulas.
- In the third part of this paper, we will prove that more generally, for the circumscription of predicates $\{p_1, ..., p_n\}$ the other predicates $\{q_1, ..., q_m\}$ being fixed, X can be the set of conjunction of clauses $P' \vee Q'$ with P' as the disjunction of p_i and Q' as the disjunction of q_j and $\neg q_k$ [Forget 1995].

2.2 PROPERTIES OF THE DEDUCTION INFERENCE

With the following we will investigate properties of the inference relation $\vdash_X$. For years, several properties for nonmonotonic inference relations have been studied. Here we focus on the most important ones. We proved that the elementary properties are verified for any X; moreover, we proved that the cumulativity is verified if the set of formulas of $\mathcal{L}$ which are not in X, is deductively closed.
In the following, A, B, C, note finite sets of formulas. In the finite case, we identifie the set $A=\{f_1, f_2, \ldots, f_n\}$ to the formula $f_1 \wedge f_2 \wedge \ldots \wedge f_n$. The following properties are all correct in infinite case, except ones which need conjonction and disjonction connectors.

Properties 1 :
For any X such that $X \neq \{True\}$ and any classical inference$\vdash$, $\vdash_X$ has the properties of :

- Supraclassicality : If $A \vdash B$ then $A \vdash_X B$
- Reflexivity : $A \vdash_X A$

- Coherence : (If $\overline{X} \neq \{True\}$) : $True \not\vdash_X False$
- Or : If $A \vdash_X C$ and $B \vdash_X C$ then $A \vee B \vdash_X C$
- Left logical equivalent :
  If $\vdash A \leftrightarrow B$, then $(A \vdash_X C \text{ iff } B \vdash_X C)$
- Right weakening :
  If $A \vdash_X B$ and $\vdash B \rightarrow C$ then $A \vdash_X C$
- Cut : If $A \vdash_X B$ and $A \wedge B \vdash_X C$ then $A \vdash_X C$
- Restricted Cumulativity :
  $A \vdash B$ iff $(A \vdash_X C$ iff $A \wedge B \vdash_X C)$

Proof :
In this proof and the following one, we use the elementary equivalencies :

$$\begin{aligned} A \vdash_X B \quad & \text{iff } (\overline{A \cup B}) \cap X = \overline{A} \cap X \\ & \text{iff } (\overline{A \cup B} - \overline{A}) \cap X = \emptyset \\ & \text{iff } (\overline{A \cup B}) \cap X \subset \overline{A} \cap X \\ & \text{iff } (\overline{A \cup B}) - \overline{A} \subset Y, \text{with} Y = F - X. \end{aligned}$$

1. Supraclassicality : If $A \vdash B$ then $A \vdash_X B$
   If $A \vdash B$, then $(\overline{A \cup B}) = \overline{A}$ for monotonicity of $\vdash$.
   Therefore $(\overline{A \cup B}) \cap X = \overline{A} \cap X$.

2. Reflexivity : $A \vdash_X A$
   Obvious for Supraclassicality and $A \vdash A$.

3. Coherence : $True \not\vdash_X False$ (if $\overline{X} \neq True$).
   If X contains a formula f which is not a tautology, we have $(\overline{True \cup False}) \cap X$ contains f (for $(True \cup False)$ is inconsistency) and $\overline{True} \cap X$ does not contain f.
   Therefore $(\overline{True \cup False}) \cap X \neq \overline{True} \cap X$

4. Or : If $A \vdash_X C$ and $B \vdash_X C$ then $A \vee B \vdash_X C$
   If $A \vdash_X C$ and $B \vdash_X C$ then
   $(\overline{A \cup C} - \overline{A}) \cap X = \emptyset$ and $(\overline{B \cup C} - \overline{B}) \cap X = \emptyset$.
   Therefore
   $(((\overline{(A \vee B) \cup C}) - \overline{A \vee B}) \cap X \subset ((\overline{A} \cup C) - \overline{A}) \cap X) \cup ((\overline{B \cup C} - \overline{B}) \cap X) = \emptyset$.

5. Left logical equivalent : If $\vdash A \leftrightarrow B$, then $(A \vdash_X C$ iff $B \vdash_X C)$
   If $\vdash A \leftrightarrow B$ then $(\overline{A \cup C}) = (\overline{B \cup C})$ and therefore $(\overline{A \cup C}) \cap X = (\overline{B \cup C}) \cap X$

6. Right weakening : If $A \vdash_X B$ and $\vdash B \rightarrow C$ then $A \vdash_X C$
   If $\vdash B \rightarrow C$ then $(\overline{A \cup C}) \subset (\overline{A \cup B})$. If $A \vdash_X B$, then $(\overline{A \cup B}) \cap X \subset \overline{A} \cap X$
   Therefore $(\overline{A \cup C}) \cap X \subset (\overline{A \cup B}) \cap X \subset \overline{A} \cap X$

7. Cut, Unit Cumulative transitivity : If $A \vdash_X B$ and $(A \wedge B) \vdash_X C$ then $A \vdash_X C$
   If $(A \wedge B) \vdash_X C$ then $(\overline{A \cup B \cup C}) \cap X \subset (\overline{A \cup B}) \cap X$
   If $A \vdash_X B$ then $(\overline{A \cup B}) \cap X \subset \overline{A} \cap X$
   Therefore $(\overline{A \cup B \cup C}) \cap X \subset \overline{A} \cap X$ (for $A \vdash_X B \wedge C$)
   Therefore, as $(\overline{A \cup C}) \subset (\overline{A \cup B \cup C})$, we have $(\overline{A \cup C}) \cap X \subset (\overline{A \cup B \cup C}) \cap X$, we have $(\overline{A \cup C}) \cap X \subset \overline{A} \cap X$
   and therefore $A \vdash_X C$

8. Restricted Cumulativity : $A \vdash B \Rightarrow (A \vdash_X C$ iff $A \wedge B \vdash_X C)$
   Obvious for suppraclassicality and if $A \vdash B$ and $A \vdash_X C$ then $(A \wedge B) \vdash_X C$.

Other important properties, linked to the cumulativity, are not verified.
For example the "and" property : If $A \vdash_X B$ *and* $A \vdash_X C$ *then* $A \vdash_X B \wedge C$ is not verified if A, B, C are propositional variables and $X = \{A \wedge B \wedge C\}$.
In the same manner the Unit Cumulative Monotony (Restricted Monotony) : If $A \vdash_X B$ and $A \vdash_X C$ then $A \wedge B \vdash_X C$ is not verified by the same $X = \{A \wedge B \wedge C\}$.

The failure of these properties can be interesting, for it enables us to hope that it will be possible to capture more than cumulative logics and more than the preferential model approach (default logics for example). On the other hand, it is possible to obtain the "cumulativity" and the "and" if the complementary of X (the set of formulas of $\mathcal{L}$ which are not in X) is deductively closed.

Properties 2 :
For any X and any classical inference $\vdash$, if the set of formulas of $\mathcal{L}$ which are not in X is deductively closed, the relation $\vdash_X$ has the properties :

- And : if $A \vdash_X B$ and $A \vdash_X C$ then $A \vdash_X B \wedge C$
- Unit Cumulative Monotony :
  If $A \vdash_X B$ and $A \vdash_X C$ then $A \wedge B \vdash_X C$
- Cumulativity (Cut + Restricted Monotony)
  $A \vdash_X B$ iff $(A \vdash_X C$ iff $A \wedge B \vdash_X C)$

Proof : First, we proved a lemma.

Lemma 1 *Let $Y = F - X$ the set of formulas of $\mathcal{L}$ which are not in X. If $\overline{Y} = Y$, then for every set A, B, C of formulas, if $(\overline{A \cup B} - \overline{A}) \subset Y$ and $(\overline{A \cup C} - \overline{A} \subset Y)$ then $(\overline{A \cup B \cup C} - \overline{A}) \subset Y$*

Proof of the lemma :
Let $f \in \overline{A \cup B \cup C} - \overline{A}$.

We have $f \in \overline{A \cup B \cup C}$ and $f \notin \overline{A}$. We will show that $f \in Y$.
As $f \in \overline{A \cup B \cup C}$, there exists $f1 \in \overline{A \cup B}$ and $f2 \in \overline{A \cup C}$ such that $f \in \overline{f1 \wedge f2}$ (the classical logic is compact).
Three cases are possible :

1. If $f1 \in \overline{A}$, then $f1 \in \overline{A \cup C}$ and, as $f2 \in \overline{A \cup C}$, we obtain $f \in \overline{A \cup C}$. Therefore $f \in Y$ by hypothesis ($\overline{A \cup C} - \overline{A} \subset Y$)
2. If, $f2 \in \overline{A}$, the proof is the same
3. If $f1 \notin \overline{A}$ and $f2 \notin \overline{A}$, then $f1 \in Y$ and $f2 \in Y$ (from hypothesis). And therefore $f \in Y$ for Y is deductively closed.

Proof of properties 2 :
The "And" property is obvious from the lemma and the definition of $\vdash_X$.
For the restricted monotonicity, we have $A \wedge B \vdash_X C$ iff $\overline{A \cup B \cup C} - \overline{A \cup B} \subset Y$.
As $(\overline{A \cup B \cup C} - \overline{A \cup B}) \subset (\overline{A \cup B \cup C} - \overline{A})$ and $(\overline{A \cup B \cup C} - \overline{A}) \subset Y$, we obtain the result from the lemma.

3 PREFERENTIAL MODEL APPROACH AND X-LOGICS

In this section, we will define a special case of preferential relations : the X-preferential relations. We prove, in the first part, that in the finite case, any preferential relation can be defined as a X-preferential relation. In the second part, we prove a representation theorem : any preferential logic is an X-logic.
The premises are the same from the first two parts; in addition the classical logic $\mathcal{L}$ is a finite propositional logic. We use classical definitions of interpretations and models
We use additional classical definitions :

- A **preferential relation** $<$ is any relation between interpretations, which is transitive (If $I < J$ and $J < K$ then $I < K$) and reflexive ($I < I$).
- An **antisymetric preferential relation**, is a preferential relation which is antisymetric (if $I < J$ and $J < I$ then $I = J$).
- If A is a set of formulas, a **minimal model of** A is an interpretation M which satisfies A ($M \vdash A$) and which is minimal, for the relation $<$ in the set of models of A (if M is a minimal of A, and if M' is a model of A such that $M' < M$, then $M < M'$).
- In a classical manner, if $<$ is a preferential relation, we define a **preferential model logic** $\vdash_<$ by $A \vdash_< B$ iff every minimal model of A satisfies B.

3.1 X-PREFERENTIAL RELATION

If X is any set of formulas of $\mathcal{L}$, the **X-preferential relation** $<_X$, is defined by :
For every interpretations I and J :
$I <_X J$

- iff $\overline{I} \cap X \subset \overline{J} \cap X$
- iff $\forall f \in X, I \vdash f \Rightarrow J \vdash f$
- iff every formula in X true for I is true for J.

It is clear that $<_X$ is a preferential relation, as defined previously. On the other hand, we prove that any preferential relation can be defined by such a X-preferential relation. For example, for the generalized closed world assumption or the sub-implication, X can be the set of positive literrals. More generally, for the circumscription of $\{p_1, ..., p_n\}$, X can be the set $\{p_1, ..., p_n\}$.

Theorem 1 *If $<$ is a preferential relation, there exists a set X of formula such that :*

A) $< \equiv <_X$ *($I < J$ iff $I <_X J$).*
B) $\vdash_< \equiv \vdash_{<X}$

Proof :
B) is obvious from A).
A) $\mathcal{F}$ is the set of formulas of the logic $\mathcal{L}$. The set X is defined by :
$X = \{f \ / \ \forall I \ \forall J, I < J \text{ and } I \vdash f \Rightarrow J \vdash f\}$
$X = \mathcal{F} \setminus \{f \ / \ \exists I, J \text{ with } I < J \text{ and } I \vdash f \text{ and } J \nvdash f\}$
(**remark** : the formulas *False* and *True* are in X)

1- $I < J \Rightarrow I <_X J$

As $I < J$, by definition, X does not contain a formula such that $I \vdash f$ and $J \nvdash f$. Therefore $\overline{I} \cap X \subset \overline{J} \cap X$.

2- $I <_X J \Rightarrow I < J$

We prove the converse : $I \not< J \Rightarrow I \not<_X J$
If $I \not< J$, there exists a formula f which :

- Opposes the relation $\vdash_{<X}$ ($I \vdash f$ and $J \nvdash f$)

- Is in X (If $M < M'$ then $M \vdash f \Rightarrow M' \vdash f$)

The formula f is built like this :

- If $K = (l_1, l_2, ..., l_n)$ is an interpretation (the l_i are the literrals true in K), we note $M(K) = (l_1 \wedge l_2 \wedge ... \wedge l_n)$
- If $I_1, I_2, ..., I_p$ is the set of interpretations which major I for $<$ ($I < I_i$)
- then $f = M(I_1) \vee M(I_2) \vee ... \vee M(I_p)$

in fact :

- By construction, f is true in every I_i (particularly in I for $I < I$), and false in the other interpretations (particularly in J).
- f is in X. Two cases are possible :
  * M' is not an I_i. As $M < M'$, by transitivity M is not an I_i. Therefore f is false in M. So f is in X.
  * M' is an I_i. By construction f is true in M' and f is in X.

3.2 REPRESENTATION THEOREM

Theorem 2 *Every preferential-model logic is an X-logic.*
(If $\vdash_<$ is a preferential logic, there exists X such that $\vdash_< \equiv \vdash_X$)
(The converse is false).

Before proving the representation theorem, some words about preferential relations. Sometimes, there is confusion with the definition of preferential logics. These prererential logics are often defined from an antisymetric relation. However, for the general case (in particular circumscription with variable predicates), the preferential relation is not antisymetric.

We encountered this problem in the proof. So, first we prove that if the preferential relation is antisymetric then the representation theorem is true (Lemma 2). Next, we prove that any preferential relation is equivalent to an antisymetric preferential relation (Lemma 3). The theorem follows obviously from these two lemmas.

Lemma 2 *If $<$ is an antisymetric preferential relation there exists X such that $\vdash_< \equiv \vdash_X$.*

Proof
We chose the same X as in the previous proof (theorem 1).
By theorem 1, we have $< \equiv <_X$ and $\vdash_< \equiv \vdash_{<X}$. Therefore, we had to prove $A \vdash_{<X} B \Leftrightarrow A \vdash_X B$.

1- $A \vdash_{<X} B \Rightarrow A \vdash_X B$.(only grounded necessary)

We had to prove that : $(\overline{A \cup B}) \cap X \subset \overline{A} \cap X$.
Let $x \in (\overline{A \cup B}) \cap X$.
Therefore x is true in every minimal model of A (for $A \vdash_{<X} B$, minimal models of A are models of B and then models of $A \cup B$). If J is a non minimal model of A, there exists I, a minimal model of A such that $I <_X J$ (grounded for finite set of models). As x is true in I (I minimal), and $I <_X J$, we have x true in J (by definition of $<_X$) Therefore, x is true in every model of A. $x \in \overline{A}$.

2- $A \vdash_X B \Rightarrow A \vdash_{<X} B$.

We proved the converse : $A \nvdash_{<X} B \Rightarrow A \nvdash_X B$.
- As $A \nvdash_{<X} B$, there exists a minimal model M of A which does not satisfy B.
- Let $I_1, ..., In$ the set of interpretations which are not inferior to M ($I_i \not<_X M$).

Remarks :

i- As $<$ is an antisymetric preferential relation, $<_X$ is an antisymetric preferential relation (for $<=<_X$).

ii- If $I <_X M$ and $I \neq M$ (strict relation), we have $M \not<_X I$ (for $<_X$ is antisymetric) and therefore I does not satisfy A (for M is a minimal model of A).

We took $x = I_1 \vee I_2 \vee ... \vee I_n$ and we proved that :

a) $x \notin \overline{A}$: By construction, x is false in M (M is not an I_i) which is a model of A.

b) $x \in \overline{A \cup B}$: If I model of $A \cup B$, I is a model of A and a model of B, therefore $I \in I_1, ..., I_n$ (only I_i's are both model of A and model of B (by remark)).

c) $x \in X$ (if $I <_X J$ and $x \in I$, then $x \in J$) : If $I <_X J$ two cases are possible :

- If $J \not<_X M$, $J \in I_1, ..., I_n$ and x is true in J (by construction).
- If $J <_X M$, we have $I <_X M$ (for $I <_X J$) and therefore $I \notin I_1, ..., I_n$, then x is false in I.

Lemma 3 *If $<$ is a preferential relation, we can build a new preferential relation $\ll$ such that :*

i) *$\ll$ is an antisymetric preferential relation.*

ii) *Any set of formula F as the same set of minimal models for the relations $<$ and $\ll$.*

Proof :
If $<$ is a preferential relation the new preferential relation $\ll$ is definied by :

a) $I \ll I$ for every I.

b) If $I < J$ and $J \not< I$ then $I \ll J$.

i) $\ll$ is an antisymetric preferential relation.
- Reflexivity : Clearly, by construction we have $I \ll I$ for every I.
- Transitivity : $\forall I, J, K, I < J$ and $J < K \Rightarrow I < K$.
  Let $I \ll J$ and $J \ll K$ we must prove that $I \ll K$.
  if $I < K$ and $I \not\ll K$ then $K < I$.
  But $I < J$ so, by transitivity of $<$, $K < J$. By definition, $J < K$ and $K < J \Rightarrow J \not\ll K$.By hypothesis, $J \ll K$ therefore $I \ll K$. $\ll$ is a transitive relation.
- Antisymetry :
  We had to prove that if $I \ll J$ and $J \ll I$ then $I = J$.
  $I \ll J \Rightarrow I < J$ and $J \not< I$ with $I \neq J$
  $J \ll I \Rightarrow J < I$ and $I \not< J$ with $I \neq J$
  therefore $I = J$. $\ll$ is an antisymetric relation.

ii) Any set of formula F as the same set of minimal models for the relations $<$ and $\ll$.
Let G a subset of F. We had to prove that : $\forall I \in G$, I is a minimal model for $< \Leftrightarrow$ I is a minimal model for $\ll$. (If I is a minimal model for the relation $<$ then $J < I \Rightarrow I < J$)
- I is a minimal model for $< \Rightarrow$ I is a minimal model for $\ll$.
  If I is a minimal model for $<$, have we $J \ll I \Rightarrow I \ll J$?
  $J \ll I \Rightarrow J < I$. But I is a minimal model for $<$ so $J < I \Rightarrow I < J$.
  Therefore $I < J$ and $J < I$.
  There is only one possibility to have $J \ll I$. It is that $I = J$. Therefore, if I is a minimal model for $<$, then $J \ll I \Rightarrow J = I$.
  Therefore I is a minimal model for $\ll$.
- I is a minimal model for $\ll \Rightarrow$ I is a minimal model for $<$.
  If I is a minimal model for $\ll$, have we $J < I \Rightarrow I < J$?
  We proved the converse : $I \not< J \Rightarrow J \not< I$.
  $I \not< J \Rightarrow I \not\ll J$. But I is a minimal model for $\ll$. therefore $J \ll I \Rightarrow I \ll J$, and $I \not\ll J \Rightarrow J \not\ll I$. therefore $I \not\ll J \Rightarrow J \not\ll I$. Then, we have $I \not\ll J$ and $J \not\ll I$ therefore, two cases are possible :
  - $I \not< J$ and $J \not< I$
  - $I < J$ and $J < I$.

  but $I \not< J$ therefore $J \not< I$. If I is a minimal model for $\ll$, then $I \not< J \Rightarrow J \not< I$ and $J < I \Rightarrow I < J$. Therefore I is a minimal model for $<$.

Proof of theorem 2 : Evident from lemmas 2 and 3.

4 X-LOGICS AND CIRCUMSCRIPTION

Previously, we defined the X-logic in a general way. In particular, for the representation theorem, even if the set X is defined in a constructive way, this set is too large to be used practicaly. On the other hand, in particular cases, it is possible to define X-logics with asmaller set X. In this section, we will establish that, for the circumscription [McCarthy 1980] of predicates $\{p_1, ..., p_n\}$, the other predicates $\{q_1, ..., q_m\}$ being fixed, X can be the set of conjunction of clauses $P' \vee Q'$ with P' disjunction of p_i and Q' disjunction of q_j and $\neg q_k$[Forget 1995].

We used a semantical definition of circumscription [Besnard-Siegel 88], based on preferential models.
If $\{p_1, \ldots, p_n\}$ are the circumscribed predicates, $\{q_1, \ldots, q_m\}$ are the fixed predicates and $\{r_1, \ldots, r_l\}$

are the variable predicates, $<_{circ}$ is definied by :

$I <_{circ} J$ iff a) $I/p_i \subset J/p_i$
b) $I/q_i = J/q_i$

(I/p_i means "the extension of p_i in I").
As for the r_i's, no condition is given, it is clear that $<_{circ}$ is an antisymetric relation iff the set of variable predicates is empty.

The circumscription entailment is noted by : $\vdash_{<_{circ}}$.
X_c is the set of conjonction of clauses formed by disjonction of predicates p_i, q_i, and $\neg q_i$.

Property 3 :
For the circumscription of predicates $\{p_1, ..., p_n\}$, the other predicates $\{q_1, ..., q_m\}$ being fixed (there are no variable predicates), the set $X_c = \{\wedge(p_i \vee q_j \vee \neg q_k)\}$ is such that : $\vdash_{<_{circ}} \equiv \vdash_X$.

The proof of this property can be found in [Forget 1995].It is based on the semantical definition of circumscription, and its scheme is the same as the one used in the proofs for theorem 1 and lemma 2.
For this proof, it is necessary that the preferential relation $<_{circ}$ is antisymetric.

In the general case of circumscription with variable predicates, $<_{circ}$ is not an antisymetric relation, so the proof is not suitable. We knew, by the theoric study presented in section 2, that there exits a transformation which enabled us to transform $<_{circ}$ in an antisymetric relation, but we have not found it yet.

5 CONCLUSIONS

This paper shows, for the finite case, that every preferential logic is an X-logic. The premises of this result can be found in [Bossu and Siegel 1982, Bossu and Siegel 1984, Bossu and Siegel 1985]. Considering the proof procedures, it is possible to use the notion of "Production field" and the algorithms linked [Siegel 1987, Katsouno and Mendelzon 1989, Katsouno and Mendelzon 1991, Boi et al. 1992]. More generally, it is possible to use ATMS algorithms. For these proof procedures, it is necessary that the choosen classical logic has a deduction theorem, as the most usual logics do. The classical logic can be a predicate logic as in [Besnard and Siegel 1988]. It can also be a modal logic, as in [Schwind and Siegel 1993, Schwind and Siegel 1994]. This preferential modal logic (hypothesis logic) can simulate a large part of default logic, and therefore it is possible to define this part of default logic as an X-logic. The possibility of defining any "classical" default logic as an X-logic remains to be investigated.

For the non-finite preferential model approach, the preferential logic can be defined as in [Bossu and Siegel 1982, Bossu and Siegel 1984, Bossu and Siegel 1985] and [Schlechta 1996]:
$A \vdash_< B$ iff there exists a set S of models of A such that :

i) every model of A is minored by an element of S
ii) if $J \in S$ and $I < J$, then $I \in S$
iii) every model of S satisfies B

For this definition, it seems that the X could be close of the X used in the proof of theorem 1.

In general cases, this paper shows also that any X-logic has the following properties : Supraclassicality, reflexivity, coherence, or, left logical equivalence, right weakening, Cut. It also proves that the "and" and the "cumulativity" are verified if the set of formulas which are not in X is deductively closed.

In this way, an interesting future research is to find a subset of nonmonotonic formalisms which can be seen as a X-logic.

References

[Besnard and Siegel 1986] Besnard and Siegel (1986). The preferential models appraoch to nonmonotonic reasonning. *Workshop Non-standard logics for Automated Reasonning.* Cordes France (Actes en [S2].[CI-4].

[Besnard and Siegel 1988] Besnard and Siegel (1988). Supposition-based logic for automates reasonning. *CADE 9 - Proc. 9th International Conference on Automated Deduction. Argonne USA.*

[Boi et al. 1992] Boi, Innocenti, Rauzy, and Siegel (1992). Production fields : a new approach to deduction problems and two algorithms for propositionnal calculus. *Revue d'Intelligence Artificielle*, 6:235–255.

[Bossu and Siegel 1982] Bossu and Siegel (1982). Nonmonotonic reasonning and databases. *Workshop Logic and databases.* CERT - Toulouse (Avtes en [S1].

[Bossu and Siegel 1984] Bossu and Siegel (1984). Nonmonotonic reasonning and databases. *Advances in Data Bases Theory*, pages 239–284. (H. Gallaire, J. Minker and J.M. Nicolas eds.) Plenum Press, New-York and London.

[Bossu and Siegel 1985] Bossu and Siegel (1985). Saturation, nonmonotonic reasonning and the close word assuption. *Artificial Intelligence*, 25:13–63.

[Forget 1995] Forget (1995). *Les logiques non monotones sont elles des X-Logiques.* Memoire de DEA, Laboratoire d'Informatique de Marseille.

[Froidevaux 1993] Froidevaux (1993). Raisonnement non-monotone et logique des defauts.

[Gardenfors and Makinson 1991] Gardenfors and Makinson (1991). Nonmonotonic inference based on expectation.

[Inoue 1992] Inoue (1992). Linear resolution for consequence finding. *Artificial Intelligence*, 56:301–353.

[Katsouno and Mendelzon 1989] Katsouno and Mendelzon (1989). A unified view of propositionnal knowledge base updates. *IJCAI 89*, pages 269–276.

[Katsouno and Mendelzon 1991] Katsouno and Mendelzon (1991). Propositionnal knowledge base revision and minimal change. *Artificial Intelligence*, 52:263–294.

[Kraus et al. 1990] Kraus, Lehmann, and Magidor (1990). Nonmonotonic reasonning, preferential models and cumulative logics. *Artificial Intelligence*, 44:167–207.

[Makinson 1989] Makinson (1989). General theory of cumulative inference. *Lecture Notes in Artificial Intelligence*, pages 1–18.

[McCarthy 1980] McCarthy (1980). Circumscription-a form of non-monotonic reasonning. *Artificial Intelligence*, 13:27–39.

[Reiter 1980] Reiter (1980). A logic for default reasonning. *Artificial Intelligence*, 13:81–132.

[Schlechta 1996] Schlechta (1996). A two-stage approach to first order default reasonning. *to appear in Fundamenta Informaticae.*

[Schwind and Siegel 1993] Schwind and Siegel (1993). Modal logic based theory for nonmonotonic reasonning. *Journal of applied non classical logic*, 3:73–92.

[Schwind and Siegel 1994] Schwind and Siegel (1994). Modal semantic for hypothesis theory. *Fundamentals or Artificia Intelligence.*

[Siegel 1981] Siegel (1981). La saturation aux secours de la non monotonie. *These de 3ieme cycle en Informatique, Universite d'Aix Marseille II.*

[Siegel 1987] Siegel (1987). Representation et utilisation de la connaissance en calcul propositionnel. *These de Doctorat d'Etat en Informatique, Universite d'Aix Marseille II.*

Representation Independence of Nonmonotonic Inference Relations

Manfred Jaeger
Max-Planck-Institut für Informatik
Im Stadtwald
66123 Saarbrücken
Germany
email: jaeger@mpi-sb.mpg.de

Abstract

A logical concept of representation independence is developed for nonmonotonic logics, including probabilistic inference systems. The general framework then is applied to several nonmonotonic logics, particularly propositional probabilistic logics. For these logics our investigation leads us to modified inference rules with greater representation independence.

1 INTRODUCTION

Entropy maximization is a rule for probabilistic inference for whose application to problems in artificial intelligence there exist several independent and very strong arguments (Grove, Halpern & Koller 1992),(Paris & Vencovská 1990). Unfortunately, though, there is a major drawback for which the maximum entropy inference rule has often been criticized: the result of the inference depends on how given information is represented.

The probably best known example used to illustrate this point is the "Life on Mars" example, a rendition of which may be given as follows: the belief that the probability for the existence of life on mars exceeds 0.6 may be expressed by the statement

$$\tau_1 :\equiv P(LoM) \geq 0.6,$$

using the vocabulary $\{LoM\}$ of propositional variables. Entropy maximization, applied to this chosen language and constraint τ_1 then yields $P(LoM) = 0.6$. Alternatively, we might choose the language $\{ALoM,PLoM\}$ containing propositional variables for "Animal Life on Mars" and "Plant Life on Mars", and express our belief by

$$\tau_2 :\equiv P(ALoM \vee PLoM) \geq 0.6.$$

Entropy maximization here yields $P(ALoM \vee PLoM) = 0.75$. Thus, the result of the maximum entropy inference rule is dependent on the choice of language, and seemingly equivalent statements yield different results.

Even though the charge of representation dependence against maximum entropy methods has been around for a long time, until recently there has neither been any precise explication of what representation independence actually is, nor a systematic investigation into the properties with regard to this property of other inference rules.

The first rigorous examination of these issues has been presented by Halpern and Koller (1995). They proceed from a definition of when two different structures, or state spaces, each equipped with a set of probability distributions, are alternative representations (namely, when on structure can be "faithfully embedded" in the other), and then call a probabilistic inference procedure representation independent when applied to the two structures it picks out corresponding preferred probability measures.

In this paper the question of representation independence will be tackled from a somewhat different perspective, and in a much wider context: first, rather than looking at specific structures and their embeddings, we here consider the purely logical question and ask: when do two sets of formulas represent the same information, and when is an inference relation defined on formulas of some language representation independent? From this syntactic perspective it is natural to extend the scope of the enquiry: a probabilistic inference rule, like entropy maximization, formally defines a nonmonotonic inference relation $\mid\!\sim$ that can be studied with respect to the same formal properties as have been investigated for nonmonotonic logics (Kraus, Lehmann & Magidor 1990), (Gabbay 1985). Conversely, a concept of representation independence,

framed entirely in terms of formulas and entailment relations, may be applied to a large class of nonmonotonic logics, not only probabilistic ones. In this paper we will develop the necessary tools for the investigation of representation independence of nonmonotonic inference relations, and take some steps towards clarifying the degree of representation (in-)dependence of existing nonmonotonic logics.

2 THE LOGICAL BACKBONE

In a similar spirit as Gabbay (1985), Kraus et al. (1990) and Makinson (1994) we will take a very general and abstract view of nonmonotonic logics. The definition we here give of what a nonmonotonic logic is puts into focus two elements that usually are either assumed only implicitly for a nonmonotonic logic, or not deemed necessary at all: the existence of a monotonic "background" entailment relation, and the possible dependence of the nonmonotonic entailment relation on the underlying vocabulary. The first of these elements will be crucial for our definition of when two knowledge bases are alternative representations; the second might be regarded as a borderline case of representation dependence.

The following definition is an adaptation of standard definitions in generalized model theory (e.g. (Ebbinghaus 1985)), tailored for the purpose at hand by the distinction of two entailment relations. Here and elsewhere we denote the powerset of X by $\mathscr{P}(X)$.

Definition 2.1 A *logic* $\mathscr{L}$ consists of

- a class of sets S, called the class of *vocabularies* of $\mathscr{L}$, which is closed under intersections and finite unions,
- for each vocabulary S a set L_S of *expressions* of $\mathscr{L}$, such that $S \subseteq S'$ implies $\mathrm{L}_S \subseteq \mathrm{L}_{S'}$.
- for each vocabulary S a relation $\vdash_S$ on $\mathscr{P}(\mathrm{L}_S) \times \mathrm{L}_S$ (the *classical entailment relation*) that is *monotonic* (i.e. $\Phi \subseteq \Phi' \subseteq \mathrm{L}_S$ and $\Phi \vdash_S \phi$ implies $\Phi' \vdash_S \phi$), and that has the *reduct property* (i.e. $\Phi \vdash_{S'} \phi$ and $S' \subseteq S$ implies $\Phi \vdash_S \phi$).

$\mathscr{L}$ is a *nonmonotonic logic* if in addition there is

- for each vocabulary S a relation $\mid\!\sim_S$ on $\mathscr{P}(\mathrm{L}_S) \times \mathrm{L}_S$ (*the nonmonotonic entailment relation*) with $\vdash_S \subseteq \mid\!\sim_S$.

Note that for the sake of simplicity we have already built the property of *supraclassicality* into the definition of a nonmonotonic logic. The concept of nonmonotonic logics is not restricted in any way by making the existence of a classical entailment relation part of the definition: whenever we have a nonmonotonic logic that lacks a natural concept of classical entailment, we can extend it to fit definition 2.1 by simply defining $\vdash_S := \emptyset$ for all S.

By virtue of the reduct property we can delete the subscript in the classical entailment relation $\vdash_S$, and simply write $\Phi \vdash \phi$, meaning that $\Phi \vdash_S \phi$ for any S with $\Phi \cup \{\phi\} \subseteq \mathrm{L}_S$. Generally, we may not expect the nonmonotonic inference relation of $\mathscr{L}$ to possess the reduct property. One counterexample is supplied by the probabilistic *center of mass* inference rule, see (Paris & Vencovská 1992) (the maximum entropy principle, on the other hand, satisfies the reduct property). Therefore, for the nonmonotonic entailment relation $\mid\!\sim_S$ we have to retain the subscript S, unless the reduct property has been established for $\mid\!\sim$ in the logic under consideration.

3 REPRESENTATIONAL VARIANTS

Before the question can be addressed, what it means for the nonmonotonic inference relation $\mid\!\sim$ to be representation independent, we have to clarify what it means for two knowledge bases Φ, Ψ to represent the same information. This will be formalized entirely in terms of the classical entailment relation $\vdash$. Throughout this section we will therefore be concerned only with the classical part of a logic.

Reconsider our introductory example. Intuitively, both τ_1 and τ_2 represent the same information with respect to the existence of life on mars. It is not immediate, however, how this intuition can be captured by a formal logical property of τ_1 and τ_2. Clearly, we are looking for something weaker than logical equivalence, because for example with $\psi :\equiv P(ALoM) < 0.3 \rightarrow P(PLoM) > 0.3$ we have $\tau_2 \vdash \psi$, but $\tau_1 \nvdash \psi$.

This, of course, is not surprising: by saying that τ_1 and τ_2 provide the same information with respect to the existence of life on mars, we do not mean to imply that τ_1 and τ_2 provide the same information regarding any other statement that can be formulated in either the vocabulary of τ_1, or the vocabulary of τ_2. Only with regard to statements that can be represented in either vocabulary will we expect the same inferences from τ_1 and τ_2. This "common ground" of the two languages can be defined by generalizing what is known as an *interpretation* in model theory (see (Hodges 1993)).

Definition 3.1 Let $\mathscr{L}$ be a logic, S, S' vocabularies of $\mathscr{L}$. An *abstract interpretation* in $\mathscr{L}$ of L_S into $L_{S'}$ with *admissibility conditions* $\Omega(f) \subset L_{S'}$ is a mapping

$$f : L_S \to L_{S'}$$

such for all $\Phi \cup \{\psi\} \subseteq L_S$:

$$\Phi \vdash_S \psi \quad \Rightarrow \quad f(\Phi) \cup \Omega(f) \vdash_{S'} f(\psi), \qquad (1)$$

where $f(\Phi) := \{f(\phi) \mid \phi \in \Phi\}$. The class of abstract interpretations in $\mathscr{L}$ is denoted $Int(\mathscr{L})$.

Example 3.2 Let $\mathscr{L}^{\text{prop}}$ be propositional logic with languages L_S^{prop}. Let S and S' be two sets of propositional variables. The extension of any function

$$f : S \to L_{S'}^{\text{prop}}.$$

to L_S^{prop} via the conditions $f(\neg\phi) \equiv \neg f(\phi)$; $f(\phi \vee \psi) \equiv f(\phi) \vee f(\psi)$ is an abstract interpretation of L_S^{prop} into $L_{S'}^{\text{prop}}$ with $\Omega(f) = \emptyset$. We call it a *propositional interpretation* and denote the class of propositional interpretations by $PI(\mathscr{L}^{\text{prop}})$.

As in this example, throughout this paper we will only encounter abstract interpretations for which we may let $\Omega(f) = \emptyset$. The admissibility conditions become relevant, for example, when we move to the standard concept of interpretations in first-order logic. Here, an atomic formula $h(x) = y \in L_S$ is mapped to a formula $\phi(x, y) \in L_{S'}$. The admissibility condition $\Omega(f)$ then would have to contain the condition that ϕ is functional, i.e. include the axiom $\forall x \exists^{=1} y \phi(x, y)$.

With abstract interpretations at our disposal, we can define what it means for two knowledge bases Φ, Ψ of a logic $\mathscr{L}$ to represent the same information with respect to a common ground – which is any language $L_{S^\bullet}$ that can be interpreted in both the language of Φ and the language of Ψ.

Definition 3.3 Let $L_{S^\bullet}$, L_{S_1}, L_{S_2} be languages in some logic $\mathscr{L}$. Let $f : L_{S^\bullet} \to L_{S_1}$ and $g : L_{S^\bullet} \to L_{S_2}$ be abstract interpretations with admissibility conditions $\Omega(f)$ and $\Omega(g)$. Let $\Phi \subseteq L_{S_1}$, $\Psi \subseteq L_{S_2}$. Φ and Ψ are called *representational variants with respect to* f *and* g, written $\Phi \overset{f,g}{\longleftrightarrow} \Psi$, iff for all $\alpha \in L_{S^\bullet}$:

$$\Phi \cup \Omega(f) \vdash_{S_1} f(\alpha) \text{ iff } \Psi \cup \Omega(g) \vdash_{S_2} g(\alpha).$$

The concept of representational varianthood being central to our development of representation independence, we shall here digress for a short while from the straightforward presentation of our main topic, and for the remainder of this section take a closer look at representational variants.

First, we turn to the question of how given knowledge bases Φ and Ψ can be proved to be representational variants, and to the even more elementary question of when a given mapping $f : L_S \to L_{S'}$ is an abstract interpretation.

For this purpose we here narrow down the class of logics to be considered to such logics in which the concepts introduced in definitions 3.1 and 3.3 as purely syntactical relations correspond to relations between semantical structures, so that questions about representational varianthood can be answered by constructing and comparing semantical structures.

Still adopting standard notions from generalized model theory, we consider logics that have a *model theoretic presentation*, and the *Boole property*. For a logic to have a model theoretic presentation means that for every vocabulary S there is defined a class of *S-structures* M, and a *satisfaction relation* $\models_S$ between S-structures and sets of S-expressions, such that $\Phi \vdash_S \psi$ iff for all S-structures M: $M \models_S \Phi$ implies $M \models_S \psi$. In propositional logic, for example, an S-structure is simply a truth assignment to the propositional variables, in first-order logic we have the usual model-theoretic S-structures. For $\Phi \subseteq L_S$ we denote by $\text{Mod}(\Phi, S)$ the class of S-structures M with $M \models \Phi$. Two classes $\mathcal{M}$ and $\mathcal{N}$ of S-structures are called *elementarily equivalent*, written $\mathcal{M} \sim \mathcal{N}$, when for all $\phi \in L_S$: $M \models \phi$ for all $M \in \mathcal{M}$ iff $N \models \phi$ for all $N \in \mathcal{N}$. A logic with a model theoretic presentation has the Boole-property iff for every $\phi, \psi \in L_S$ there exist expressions $\neg\phi, \phi \vee \psi \in L_S$ such that for every S-structure M: $M \models \neg\phi$ iff $M \not\models \phi$, and $M \models \phi \vee \psi$ iff $M \models \phi$ or $M \models \psi$.

Logics with a model-theoretic presentation and the Boole property permit us to apply the usual rules of classical logic for the manipulation of boolean connectives with respect to the classical entailment relation. Particularly, in any such logic the deduction theorem is valid: $\Phi \cup \{\phi\} \vdash \psi$ iff $\Phi \vdash \phi \to \psi$ ($\phi \to \psi$ being, of course, and abbreviation for $\neg\phi \vee \psi$).

We now obtain the following connection between S-structures and interpretations:

Lemma 3.4 Let $\mathscr{L}$ be a logic with a model theoretic presentation, $f : L_S \to L_{S'}$, $\Omega(f) \subset L_{S'}$. If for every S'-structure M with $M \models \Omega(f)$ there exists an S-structure N such that for all $\phi \in L_S$:

$$M \models f(\phi) \text{ iff } N \models \phi \qquad (2)$$

then f is an abstract interpretation with admissibility conditions $\Omega(f)$.

When $\mathscr{L}$ has the Boole-property, and f is compat-

ible with boolean connectives, i.e. $f(\neg\phi) \equiv \neg f(\phi)$, $f(\phi \vee \psi) \equiv f(\phi) \vee f(\psi)$ $(\phi, \psi \in \mathrm{L}_S)$, then the converse also holds: if f is an abstract interpretation with admissibility conditions $\Omega(f)$ then for each $M \models \Omega(f)$ there exists an S-structure N with (2).

Proof: For the first part of the lemma let f and $\Omega(f)$ be given. Let $\Phi \cup \{\psi\} \subseteq \mathrm{L}_S$ with $\Phi \vdash \psi$. If $M \not\models f(\Phi)$ for all S'-structures M with $M \models \Omega(f)$ then (1) is trivially satisfied. Now suppose that $M \models f(\Phi) \cup \Omega(f)$, and let N be an S-structure satisfying (2). Then $N \models \Phi \cup \{\psi\}$, hence $M \models f(\psi)$, and therefore $f(\Phi) \cup \Omega(f) \vdash f(\psi)$.

For the second part assume that f is an abstract interpretation with admissibility conditions $\Omega(f)$ that is compatible with boolean connectives. Let $M \models \Omega(f)$ be given, and define $\Phi := \{\phi \in \mathrm{L}_S \mid M \models f(\phi)\}$. Assume that there does not exist an S-structure N with $N \models \Phi$. Then $\Phi \vdash \phi \wedge \neg\phi$ for arbitrary $\phi \in \mathrm{L}_S$, and by (1) $f(\Phi) \cup \Omega(f) \vdash f(\phi) \wedge \neg f(\phi)$, contradicting the assumption that $M \models f(\Phi) \cup \Omega(f)$. □

When for an abstract interpretation $f : \mathrm{L}_S \to \mathrm{L}_{S'}$ and any S'-structure M an S-structure N with (2) exists, then f is said to *define associated structures.* The mapping that assigns an associated structure N to S'-structures M is denoted by $\bar{f}$. Generally, this mapping will not be uniquely determined by condition (2), and $\bar{f}(M)$ can be any (usually canonical) selection of one of the associated S-structures for M.

Example 3.5 Let $S = \{A_1, \ldots, A_k\}$, $S' = \{B_1, \ldots, B_l\}$ be propositional vocabularies, $f : \mathrm{L}_S \to \mathrm{L}_{S'}$; $f(A_i) :\equiv \phi_{A_i}$ a propositional interpretation.

Let $M : S' \to \{\mathit{true}, \mathit{false}\}$ be a truth assignment. Then the unique truth assignment $\bar{f}(M) : S \to \{\mathit{true}, \mathit{false}\}$ for which (2) holds is defined by

$$\bar{f}(M)(A_i) := M(\phi_{A_i}) \quad (A_i \in S).$$

A criterion for representational varianthood now is given by the following lemma.

Lemma 3.6 Let $\mathscr{L}$ be a logic with a model theoretic presentation. Let $f : \mathrm{L}_{S^*} \to \mathrm{L}_{S_1}$, $g : \mathrm{L}_{S^*} \to \mathrm{L}_{S_2}$ be abstract interpretations that define associated structures $\bar{f}(\cdot), \bar{g}(\cdot)$. Let $\Phi \subseteq \mathrm{L}_{S_1}$, $\Psi \subseteq \mathrm{L}_{S_2}$. Then $\Phi \overset{f,g}{\longleftrightarrow} \Psi$ iff

$$\bar{f}(\mathrm{Mod}(\Phi \cup \Omega(f), S_1)) \sim \bar{g}(\mathrm{Mod}(\Psi \cup \Omega(g), S_2)).$$

The proof is immediate from the definitions.

Example 3.7 Let S_1 be the set of propositional variables *LoM* (Life on Mars), *LoE* (Life on Earth), *LiSS* (Life in Solar System), *IntL* (Intelligent Life), and *EalL* (Earth-like Life). Let S_2 consist of *ALoM* (Animal Life on Mars), *PLoM* (Plant Life on Mars), *LiSS* (Life in Solar System), and *HL* (Human Life).

Let

$$\phi \equiv (\mathit{LoM} \to \mathit{LiSS} \wedge \neg \mathit{IntL}) \wedge (\mathit{LoE} \to \mathit{EalL}) \quad \in \mathrm{L}_{S_1}$$
$$\psi \equiv (\mathit{ALoM} \vee \mathit{PLoM}) \to (\mathit{LiSS} \wedge \neg \mathit{HL}) \quad \in \mathrm{L}_{S_2}$$

Here we think of the propositional variables rather as relation symbols, designating the set of all objects that are "life on mars", "intelligent life", and so on, rather than as propositions "there exists life on mars", etc.

The only propositional variable that the two vocabularies have in common is *LiSS*. Still we can extract the same information about the relation of life on mars, life in the solar system, and human life from the two formulas. To make this precise, consider the propositional interpretations f, g of $S^* := \{A, B, C\}$ into S_1 and S_2, respectively defined by

$$f: \quad A \mapsto \mathit{LoM}, \quad B \mapsto (\mathit{IntL} \wedge \mathit{EalL}), \quad C \mapsto \mathit{LiSS};$$
$$g: \quad A \mapsto (\mathit{ALoM} \vee \mathit{PLoM}), \quad B \mapsto \mathit{HL}, \quad C \mapsto \mathit{LiSS}.$$

We may now use lemma 3.6 to show that $\phi \overset{f,g}{\longleftrightarrow} \psi$. For this example, and for propositional logic in general, this is particularly simple, because here for any classes $\mathcal{M}, \mathcal{N}$ of structures $\mathcal{M} \sim \mathcal{N}$ is equivalent to $\mathcal{M} = \mathcal{N}$. In the current example it is readily verified that both $\bar{f}(\mathrm{Mod}(\phi, S_1))$ and $\bar{g}(\mathrm{Mod}(\psi, S_2))$ are equal to the set $\{(t,f,t), (f,f,t), (f,t,t), (f,t,f), (f,f,f)\}$ of truth assignments to the variables (A, B, C), which corresponds to the set of models of the S^*-formula $A \to C \wedge \neg B$.

The two formulas ϕ and ψ in this example differ in one aspect that we will later find to be quite essential: while ψ actually is the formula $g(A \to C \wedge \neg B)$, it is not the case that $\phi \equiv f(A \to C \wedge \neg B)$. Moreover, ϕ is not even equivalent to any formula $f(\alpha)$ with $\alpha \in \mathrm{L}_{S^*}$, because ϕ makes a nontrivial statement about the propositional variable *LoE*, which does not appear in the range of f.

The following example is useful to caution us against expecting more from the relation $\Phi \overset{f,g}{\longleftrightarrow} \Psi$ than there actually is in it. For this example and further use below we introduce the notation $\mathit{id}(\mathrm{L}_S)$ to denote the "identity interpretation" of L_S, i.e. the abstract interpretation $\mathit{id} : \mathrm{L}_S \to \mathrm{L}_S$ with $\mathit{id}(\alpha) \equiv \alpha$ for all $\alpha \in \mathrm{L}_S$, and $\Omega(\mathit{id}) = \emptyset$.

Example 3.8 Let $S^* = S_1 = \{A\}$, $S_2 = \{B, C\}$ be propositional vocabularies, $f = id(\mathrm{L}_{S^*})$, $g : \mathrm{L}_{S^*}^{\mathrm{prop}} \to \mathrm{L}_{S_2}^{\mathrm{prop}}$; $g(A) :\equiv B \vee C$ be propositional interpretations. Let $\phi :\equiv A$ and $\psi :\equiv B \wedge C$. Then $\phi \overset{f,g}{\longleftrightarrow} \psi$. It is not true, however, that also $\neg\phi \overset{f,g}{\longleftrightarrow} \neg\psi$, because $\neg\phi \vdash f(\neg A)$, whereas $\neg\psi \not\vdash \neg(B \vee C) \equiv g(\neg A)$.

By a similar example it can also be shown that $\phi \overset{f,g}{\longleftrightarrow} \psi$ and $\phi' \overset{f,g}{\longleftrightarrow} \psi'$ does not imply $\phi \wedge \phi' \overset{f,g}{\longleftrightarrow} \psi \wedge \psi'$.

Interestingly, however, in any logic with a model-theoretic presentation and the Boole property $\phi \overset{f,g}{\longleftrightarrow} \psi$ and $\phi' \overset{f,g}{\longleftrightarrow} \psi'$ does imply $\phi \vee \phi' \overset{f,g}{\longleftrightarrow} \psi \vee \psi'$, because for $\alpha \in \mathrm{L}_{S^*}$ we then have $\phi \vee \phi' \vdash f(\alpha)$ iff $\phi \vdash f(\alpha)$ and $\phi' \vdash f(\alpha)$ iff $\psi \vdash g(\alpha)$ and $\psi' \vdash g(\alpha)$ iff $\psi \vee \psi' \vdash g(\alpha)$.

4 REPRESENTATION INDEPENDENCE

Having defined what it means that Φ and Ψ represent the same information (with respect to a commonly interpretable language L_{S^*}), we can address our main issue, and define representation (in)dependence of $\vdash$.

Now that we will be dealing with the nonmonotonic entailment relation of a logic, somewhat greater care than in the preceding section has to be taken with respect to specifying what vocabulary is being assumed as defining the background language in which a knowledge base is to be evaluated. Basically, this just means that we have to more conscientiously use the properly indexed entailment operator $\vdash_S$. In addition, it will be convenient to also add the specification of a vocabulary S with $\Phi \subseteq \mathrm{L}_S$ as an external attribute to a knowledge base. We therefore define the class of ***knowledge bases*** of $\mathscr{L}$ as the set of pairs

$$KB(\mathscr{L}) = \{(\Phi, S) \mid \Phi \subseteq \mathrm{L}_S;\ S \text{ vocabulary}\}.$$

Whenever the specification of the vocabulary S can be dispensed with, either because the logic under consideration is known to have the reduct property for $\vdash$, or because the underlying vocabulary is clear or irrelevant in the given context, we will continue to simply write Φ for (Φ, S).

The general thrust of the definition of representation independence should be quite obvious: the nonmonotonic logic $\mathscr{L}$ is representation independent (r.i.) if, whenever (Φ, S_1) and (Ψ, S_2) represent the same information about L_{S^*}, then $\mathscr{L}$ also permits the same nonmonotonic inferences from (Φ, S_1) and (Ψ, S_2) about L_{S^*}, i.e. if $\Phi \overset{f,g}{\longleftrightarrow} \Psi$, then for all $\alpha \in \mathrm{L}_{S^*}$:

$$\Phi \cup \Omega(f) \vdash_{S_1} f(\alpha) \text{ iff } \Psi \cup \Omega(g) \vdash_{S_2} g(\alpha). \quad (*)$$

This is essentially the definition of r.i. that we will give below. However, a few complications are inevitable: we can not hope to obtain $(*)$ for any interesting nonmonotonic logic for arbitrary Φ, Ψ, f, and g. Usually, we will have to look a little closer at how Φ, Ψ, f, and g can interact in a specific logic, and try to obtain only qualified statements of the form $(*)$.

There are basically two ways to impose restrictions on the statements of this form: we may consider only a special set of abstract interpretations, not arbitrary ones, and we may restrict the set of admissible knowledge bases Φ and Ψ, where these sets may also depend on the interpretations considered.

The first of these limitations is really inevitable: whenever we consider a specific logic $\mathscr{L}$, we will be concerned with certain classes of natural interpretations, e.g. in the context of propositional logic the propositional interpretations of example 3.2, and not with any abstract interpretation in the sense of definition 3.1. Restrictions of the second type, on the other hand, usually mean a real compromising of the strength of the results obtained.

Definition 4.1 Let $\mathscr{L}$ be a nonmonotonic logic. Let

$$\boldsymbol{F} \subseteq Int(\mathscr{L}) \times Int(\mathscr{L}) \times KB(\mathscr{L}) \times KB(\mathscr{L}).$$

$\mathscr{L}$ is ***representation independent with respect to*** $\boldsymbol{F}$, iff $(f, g, (\Phi, S_1), (\Psi, S_2)) \in \boldsymbol{F}$ and $\Phi \overset{f,g}{\longleftrightarrow} \Psi$ implies

$\forall \alpha \in \mathrm{L}_{S^*}$:

$$\Phi \cup \Omega(f) \vdash_{S_1} f(\alpha) \text{ iff } \Psi \cup \Omega(g) \vdash_{S_2} g(\alpha), \quad (3)$$

where L_{S^*} is the common domain of f and g.

Note that in this definition the condition $(f, g, (\Phi, S_1), (\Psi, S_2)) \in \boldsymbol{F}$ in itself does not guarantee that f and g are defined on the same language L_{S^*}. However, this is implicit in the additional condition $\Phi \overset{f,g}{\longleftrightarrow} \Psi$.

The following examples show how the general concept of representation independence of definition 4.1 subsumes special properties previously considered in the literature.

Example 4.2 The reduct property for the nonmonotonic entailment relations $\vdash_S$ may be represented in the framework of definition 4.1 as follows. Let $\boldsymbol{F}_{RP}$ contain all tuples of the form $(f, f, (\Phi, S_1), (\Phi, S_2))$ where $f = id(\mathrm{L}_S)$ for some L_S with $\Phi \subseteq \mathrm{L}_S$. For

any such tuple $\Phi \overset{f,f}{\longleftrightarrow} \Phi$ then holds, and nonmonotonic entailment in $\mathscr{L}$ has the reduct property iff $\mathscr{L}$ is r.i. with respect to F_{RP}.

Example 4.3 One of the simplest changes of representation one can devise is by renaming the nonlogical symbols in an expression. Since definition 2.1 does not require that the expressions in L_S will actually be strings constructed from elements of S, we may not have a feasible concept of renaming for every logic in the sense of definition 2.1. Therefore, assume that $\mathscr{L}$ is a logic in which each bijection $f : S \to S'$ of vocabularies (respecting such attributes as sort and arity of symbols, if these exist) induces a bijection $f : L_S \to L_{S'}$ of the corresponding languages such that f is an abstract interpretation of L_S in $L_{S'}$, and f^{-1} is an abstract interpretation of $L_{S'}$ in L_S with $\Omega(f) = \Omega(f^{-1}) = \emptyset$ (this just means that the classical part or $\mathscr{L}$ has the *renaming property* (Ebbinghaus 1985)).

Now let $\boldsymbol{F}_R$ contain all tuples of the form $(f, g, (\Phi, S), (g(\Phi), S'))$ where $f = id(L_S)$, and $g : S \to S'$ is a renaming. By the renaming property of $\vdash$ any such tuple already satisfies $\Phi \overset{f,g}{\longleftrightarrow} g(\Phi)$, and nonmonotonic entailment in $\mathscr{L}$ is invariant under renaming iff $\mathscr{L}$ is r.i. with respect to $\boldsymbol{F}_R$.

Example 4.4 Another basic form of changing representation is by choosing a logically equivalent knowledge base (in the sense of the classical entailment relation) in the same language. When such a change does not affect the set of formulas entailed by $\vdash$ a nonmonotonic logic $\mathscr{L}$ is said to have the property of *left logical equivalence* (cf. (Makinson 1994)).

This property, too, can be very easily captured as a form of representation independence in the sense of definition 4.1. Let $\boldsymbol{F}_{LLE}$ contain all tuples of the form $(f, f, (\Phi, S_1), (\Psi, S_2))$ where $f = id(L_S)$ for some S with $\Phi, \Psi \subseteq L_S$. The condition $\Phi \overset{f,f}{\longleftrightarrow} \Psi$ then means that Φ and Ψ are logically equivalent, and $\mathscr{L}$ is r.i. with respect to $\boldsymbol{F}_{LLE}$ iff $\mathscr{L}$ satisfies left logical equivalence.

Note that $\boldsymbol{F}_{RP} \subseteq \boldsymbol{F}_{LLE}$, so that left logical equivalence as defined here implies the reduct property.

Definition 4.1, once again, is formulated on a purely syntactical level. Pursuing our endeavor to supplement these syntactical concepts with corresponding semantical notions, we shall now describe a semantical approach to representation independence for a certain class of nonmonotonic logics.

As previously we had to narrow down the class of logics to be considered to those in which classical entailment is defined by a model-theoretic presentation, we now have to focus on such logics in which nonmonotonic entailment, too, has a semantical background. A large and natural such class is comprised of those logics in which nonmonotonic entailment has a *preferential model semantics* (Shoham 1987). We here generalize the notion of preferential models in one aspect, for which we need the following preparatory definition.

Definition 4.5 Let $\mathscr{L}$ be a logic with a model-theoretic presentation, S a vocabulary, and $\mathcal{S}$ the class of S-structures. We say that $\mathcal{S}$ is *locally ordered* iff on every subset $\mathcal{S}' \subseteq \mathcal{S}$ a unique partial order is defined. $\mathcal{S}$ is said to be *globally ordered* iff $\mathcal{S}$ is locally ordered, and for each $\mathcal{S}' \subseteq \mathcal{S}$ the order on $\mathcal{S}'$ is given by the restriction to $\mathcal{S}'$ of the order on $\mathcal{S}$. We denote by $\lfloor \mathcal{S}' \rfloor$ the set of elements of $\mathcal{S}'$ that are minimal in the ordering of $\mathcal{S}'$.

Definition 4.6 Let $\mathscr{L}$ be a nonmonotonic logic with a model-theoretic presentation. $\mathscr{L}$ has a *local preferential model semantics* iff for each vocabulary S the class $\mathcal{S}$ is locally ordered, and for each $\Phi \cup \{\phi\} \subseteq L_S$: $\Phi \vdash_S \phi$ iff $M \models \phi$ for all $M \in \lfloor Mod(\Phi, S) \rfloor$. $\mathscr{L}$ has a *global preferential model semantics* iff $\mathscr{L}$ has a local preferential model semantics defined by a global order on each $\mathcal{S}$.

Hence, the usual notion of preferential model semantics corresponds to a global preferential model semantics as defined here. The generalization to local preferential model semantics is necessary in order to accommodate certain probabilistic logics (cf. section 5.3), and may also deserve some attention in other contexts. We now obtain the following criterion for representation independence.

Lemma 4.7 Let $\mathscr{L}$ be a logic with a local preferential model semantics, let $\boldsymbol{F}$ be as in definition 4.1 such that all f and g that appear in a tuple of $\boldsymbol{F}$ define associated structures. Then $\mathscr{L}$ is r.i. with respect to $\boldsymbol{F}$ iff for every tuple $(f, g, (\Phi, S_1), (\Psi, S_2)) \in \boldsymbol{F}$ with $\Phi \overset{f,g}{\longleftrightarrow} \Psi$

$$\bar{f}(\lfloor Mod(\Phi \cup \Omega(f), S_1) \rfloor) \sim \bar{g}(\lfloor Mod(\Psi \cup \Omega(g), S_2) \rfloor).$$

The proof is immediate from the definitions.

5 CASE STUDIES

In this section the general framework developed so far is applied to a selection of specific nonmonotonic logics.

5.1 CLOSED WORLD ASSUMPTION AND NEGATION AS FAILURE

The closed world assumption is a very general form of nonmonotonic reasoning that is not restricted to any specific logical framework. Any classical logic $\mathscr{L}$ that has a model theoretic presentation and the Boole property (actually, only any feasible concept of negation is required) can be extended to a nonmonotonic logic $\mathscr{L}^{\text{cwa}}$ by the definition $\Phi \vdash \phi$ iff $\Phi \nvdash \neg\phi$ (or $\Phi \vdash \neg\phi \wedge \phi$). It is obvious that $\mathscr{L}^{\text{cwa}}$ is r.i. in a very strong sense, namely with respect to $\boldsymbol{F} = \mathit{Neg} \times \mathit{Neg} \times \mathit{KB}(\mathscr{L}^{\text{cwa}}) \times \mathit{KB}(\mathscr{L}^{\text{cwa}})$, where $\mathit{Neg} \subseteq \mathit{Int}(\mathscr{L}^{\text{cwa}})$ is the class of abstract interpretations f with $f(\neg\alpha) \equiv \neg f(\alpha)$ for all α.

In logic programming an approximation of the closed world assumption may be implemented by the negation as failure rule (Clark 1978). Here we lose completely the representation independence of the closed world assumption; in fact, negation as failure does not even satisfy left logical equivalence: from the two logic programs

$$P: \begin{array}{l} Rf(x) \leftarrow Rf(x) \\ Qa \end{array} \quad \text{and} \quad P': \begin{array}{l} Rf(x) \leftarrow Rx \\ Qa \end{array}$$

the same literals are provable (namely, just Qa), so that in the sense of logic programming they are logically equivalent. However, for any literal $L \equiv Rf^n(a)$ $(n \geq 1)$, an attempted proof of L from P will not terminate, while it will fail from P'. Hence we get $P \nvdash \neg Rf^n(a)$, but $P' \vdash \neg Rf^n(a)$ for all $n \geq 1$.

5.2 RATIONAL CLOSURE

Lehmann's (1989) rational closure and Pearl's (1990) system Z are two systems for evaluating *conditional knowledge bases* that for finitary knowledge bases have been found to be equivalent (Pearl 1990). Pearl (1990) also has shown that this logic can be defined by a preferential model semantics using *κ-rankings* going back to Spohn (1990). We here use this last approach for defining the (finitary) rational closure logic $\mathscr{L}^{\text{rc}}$.

The class of vocabularies of $\mathscr{L}^{\text{rc}}$ consists of all finite propositional vocabularies. An S-expression is a *conditional* $\phi \rightsquigarrow \psi$ with $\phi, \psi \in \mathrm{L}_S^{\text{prop}}$. An S-structure is a *ranking function*

$$\kappa : W_S \to \mathbf{N} \cup \{\infty\}$$

where $W_S = \{w_1, \ldots, w_n\}$ is the set of the $n = 2^{|S|}$ truth assignments for S. A ranking function κ satisfies a conditional $\phi \rightsquigarrow \psi$ iff

$$\mathit{min}\{\kappa(w) \mid w \models \phi \wedge \psi;\ w \in W_S\} < \mathit{min}\{\kappa(w) \mid w \models \phi \wedge \neg\psi;\ w \in W_S\},$$

using the conventions $\mathit{min}\emptyset = \infty$ and $\infty < \infty$. The class $\mathcal{S}$ of ranking functions for S is globally ordered by: $\kappa < \kappa'$ iff $\forall w \in W_S :\ \kappa(w) \leq \kappa'(w)$ and $\exists w \in W_S :\ \kappa(w) < \kappa'(w)$. Nonmonotonic entailment in $\mathscr{L}^{\text{rc}}$ then is defined by the preferential model semantics with respect to this ordering.

The definition of $\mathscr{L}^{\text{rc}}$ in one important aspect is of a somewhat different nature than those of many other nonmonotonic logics: in $\mathscr{L}^{\text{rc}}$ the nonmonotonic entailment relation of the logic, i.e. the entailment of conditionals $\phi \rightsquigarrow \psi$ from a set Φ of given conditionals, is not directly an image of the nonmonotonic commonsense reasoning that the logic wants to describe. Such reasoning, rather, is represented by the object-language operator $\rightsquigarrow$, and a commonsense inference of ψ from a given fact ϕ and a set Φ of default rules is modeled in the formal logic by the nonmonotonic entailment of $\phi \rightsquigarrow \psi$ from Φ.

$\mathscr{L}^{\text{rc}}$ satisfies the reduct property for nonmonotonic entailment, so that most of the time a reference to the underlying vocabulary here is unnecessary.

Lehmann & Magidor (1992) have observed that rational closure is invariant under renaming. We here extend this result for $\mathscr{L}^{\text{rc}}$ and show that this logic is r.i. in a more general sense.

We consider abstract interpretations $f^* : \mathrm{L}_S^{\text{rc}} \to \mathrm{L}_{S'}^{\text{rc}}$ obtained by extending propositional interpretations $f \in \mathit{PI}(\mathscr{L}^{\text{prop}})$ to L_S^{rc} via

$$f^*(\phi \rightsquigarrow \psi) :\equiv f(\phi) \rightsquigarrow f(\psi). \tag{4}$$

The following lemma tells us that f^* really is an abstract interpretation for $\mathscr{L}^{\text{rc}}$.

Lemma 5.1 Let $f \in \mathit{PI}(\mathscr{L}^{\text{prop}})$, $f : \mathrm{L}_S^{\text{prop}} \to \mathrm{L}_{S'}^{\text{prop}}$. Then $f^* : \mathrm{L}_S^{\text{rc}} \to \mathrm{L}_{S'}^{\text{rc}}$ as defined by (4) is an abstract interpretation with $\Omega(f^*) = \emptyset$.

Proof: We use lemma 3.4 and show that f^* defines associated structures. Let $\kappa : W_{S'} \to \mathbf{N} \cup \{\infty\}$ be an arbitrary S'-structure. An S-structure $\bar{f}^*(\kappa)$ is defined by letting for $w \in W_S$:

$$\bar{f}^*(\kappa)(w) := \mathit{min}\{\kappa(w') \mid w = \bar{f}(w');\ w' \in W_{S'}\}.$$

For every $\alpha \in \mathrm{L}_S^{\text{prop}}$ we then have

$$\begin{aligned} &\mathit{min}\{\bar{f}^*(\kappa)(w) \mid w \models \alpha;\ w \in W_S\} \\ &\quad = \mathit{min}\{\kappa(w') \mid w = \bar{f}(w');\ w \models \alpha;\ w' \in W_{S'}\} \\ &\quad = \mathit{min}\{\kappa(w') \mid w' \models f(\alpha);\ w' \in W_{S'}\} \end{aligned}$$

(note that in the special case that α only is true in $w \in W_S$ for which $\{w' \in W_{S'} \mid \bar{f}(w') = w\} = \emptyset$ we receive the value ∞ for the terms in this identity).

It now follows immediately that $\kappa \models f^*(\alpha \rightsquigarrow \beta)$ iff $\bar{f}^*(\kappa) \models \alpha \rightsquigarrow \beta$ for all $\alpha \rightsquigarrow \beta \in \mathrm{L}^{\mathrm{rc}}_S$, and hence that f^* is an abstract interpretation. □

The class of abstract interpretations defined by (4) is denoted $PI(\mathscr{L}^{\mathrm{rc}})$. For simplicity, in the sequel we will no longer distinguish in our notation an interpretation $f \in PI(\mathscr{L}^{\mathrm{rc}})$ from its underlying propositional interpretation $f \in PI(\mathscr{L}^{\mathrm{prop}})$. The distinction is implicit in the argument of the function, which may be either a propositional formula, or a conditional, and in the case of $\bar{f}$, a truth assignment, or a ranking function.

Letting

$$\boldsymbol{F}_R := \{(f, g, \Phi, \Psi) \mid f, g \in PI(\mathscr{L}^{\mathrm{rc}}); \Phi \subseteq f(\mathrm{L}^{\mathrm{rc}}_{S^\bullet}), \Psi \subseteq g(\mathrm{L}^{\mathrm{rc}}_{S^\bullet})\},$$

we now obtain the following result about representation independence of $\mathscr{L}^{\mathrm{rc}}$.

Theorem 5.2 $\mathscr{L}^{\mathrm{rc}}$ is r.i. with respect to $\boldsymbol{F}_R$.

Proof: The theorem is proved using lemma 4.7 by showing that

$$\bar{f}(\lfloor \mathrm{Mod}(\Phi, S_1) \rfloor) = \bar{g}(\lfloor \mathrm{Mod}(\Psi, S_2) \rfloor) \qquad (5)$$

for $(f, g, \Phi, \Psi) \in \boldsymbol{F}_R$ with $\Phi \overset{f,g}{\longleftrightarrow} \Psi$.

For this, let $\kappa_1 \in \lfloor \mathrm{Mod}(\Phi, S_1) \rfloor$. We define a ranking function κ_2 on W_{S_2} by

$$\kappa_2(w) := \bar{f}(\kappa_1)(\bar{g}(w)) \quad (w \in W_{S_2}).$$

It now has to be verified that $\bar{g}(\kappa_2) = \bar{f}(\kappa_1)$, and that $\kappa_2 \in \lfloor \mathrm{Mod}(\Psi, S_2) \rfloor$.

Let $w \in W_{S^\bullet}$. Then

$$\begin{aligned}\bar{g}(\kappa_2)(w) &= min\{\kappa_2(w') \mid w = \bar{g}(w');\ w' \in W_{S_2}\} \\ &= min\{\bar{f}(\kappa_1)(\bar{g}(w')) \mid w = \bar{g}(w');\ w' \in W_{S_2}\} \\ &= min\{\bar{f}(\kappa_1)(w) \mid w = \bar{g}(w');\ w' \in W_{S_2}\}.\end{aligned}$$

When $\bar{g}^{-1}(w) \neq \emptyset$ we clearly obtain $\bar{f}(\kappa_1)(w)$ on the right hand side of this equation. Else, we get $min\,\emptyset = \infty$. To see that in this case $\bar{f}(\kappa_1)(w) = \infty$ also must hold, let ω be a propositional S^*-formula that is true only under the truth assignment w. From $\bar{g}^{-1}(w) = \emptyset$ it follows that $g(\omega)$ is not satisfiable in $\mathscr{L}^{\mathrm{prop}}$. Hence, in $\mathscr{L}^{\mathrm{rc}}$: $\vdash g(\omega) \rightsquigarrow \mathit{false}$. From $\Phi \overset{f,g}{\longleftrightarrow} \Psi$ it then follows that $\Phi \vdash f(\omega) \rightsquigarrow \mathit{false}$, meaning that

$$\begin{aligned}\infty &= min\{\kappa_1(w') \mid w' \models f(\omega);\ w' \in W_{S_1}\} \\ &= min\{\kappa_1(w') \mid w = \bar{f}(w');\ w' \in W_{S_1}\} \\ &= \bar{f}(\kappa_1)(w).\end{aligned}$$

From $\Psi \subseteq g(\mathrm{L}^{\mathrm{rc}}_{S^\bullet})$ it immediately follows that $\kappa_2 \models \Psi$: for every conditional $g(\alpha) \rightsquigarrow g(\beta) \in \Psi$ $(\alpha, \beta \in \mathrm{L}^{\mathrm{prop}}_{S^\bullet})$ trivially $\Psi \vdash g(\alpha) \rightsquigarrow g(\beta)$, and hence $\Phi \vdash f(\alpha) \rightsquigarrow f(\beta)$. Thus, $\bar{f}(\kappa_1) = \bar{g}(\kappa_2) \models \alpha \rightsquigarrow \beta$, and $\kappa_2 \models g(\alpha) \rightsquigarrow g(\beta)$.

Now assume that there exists $\kappa_2' \in \mathrm{Mod}(\Psi, S_2)$ with $\kappa_2' < \kappa_2$. From the special form of κ_2 (namely, the fact that $w_1, w_2 \in W_{S_2}$ with $\bar{g}(w_1) = \bar{g}(w_2)$ have the same rank in κ_2) it follows that then $\bar{g}(\kappa_2') < \bar{g}(\kappa_2)$. In the same manner as κ_2 was defined we now construct $\kappa_1' \in \mathrm{Mod}(\Phi, S_1)$ with $\bar{f}(\kappa_1') = \bar{g}(\kappa_2')$. By the special form of κ_1', and from $\bar{f}(\kappa_1') < \bar{f}(\kappa_1)$ it now follows that $\kappa_1' < \kappa_1$, contradicting the minimality of κ_1. Thus, $\kappa_2 \in \lfloor \mathrm{Mod}(\Psi, S_2) \rfloor$, which concludes the proof that $\bar{f}(\lfloor \mathrm{Mod}(\Phi, S_1) \rfloor) \subseteq \bar{g}(\lfloor \mathrm{Mod}(\Psi, S_2) \rfloor)$. By symmetry, the converse inclusion holds as well, which proves (5) □

Results similar to theorem 5.2 also hold for default logic (Reiter 1980) and Boutelier's (1994) conditional logic CT4O, among others. In each of these cases representation independence holds with respect to a set $\boldsymbol{F}$ whose definition includes the condition that $(f, g, \Phi, \Psi) \in \boldsymbol{F}$ only when $\Phi \subseteq f(\mathrm{L}_{S^\bullet})$ and $\Psi \subseteq g(\mathrm{L}_{S^\bullet})$. This condition (which we will also encounter in the following section) so far we have only been able to dispense with in case of the simple logic $\mathscr{L}^{\mathrm{cwa}}$. It may be possible in some cases to relax this condition when at the same time the class of interpretations admitted in $\boldsymbol{F}$ is further restricted. One natural restriction one may consider for interpretations f is to demand that for f the converse implication also holds in (1) (this is the syntactic analogue to Halpern and Koller's (1995) condition of "faithfulness" for embeddings of state spaces).

5.3 PROPOSITIONAL PROBABILISTIC LOGIC

We now return to our starting point: probabilistic inference rules, specifically a type of inference rules that have been studied extensively (e.g. (Diaconis & Zabell 1982), (Paris & Vencovská 1992), (Halpern & Koller 1995)), and which might be designated *measure selection inferences.*

In order to fit these rules into our framework, we first have to present them as nonmonotonic logics in the sense of definition 2.1. The classical parts of each of these logics will be the same and similar to several previously defined formalisms (e.g. (Nilsson 1986), (Frisch & Haddaway 1994)). We denote it by $\mathscr{L}^{\mathrm{pp}}$, and define it as follows.

The class of vocabularies of $\mathscr{L}^{\mathrm{pp}}$ is the class of all

finite sets S of propositional variables. To define the language $\mathrm{L}_S^{\mathrm{pp}}$, we first inductively define an *S-term* to be either a constant symbol for a rational number, a *probability expression* $P(\phi)$, where $\phi \in \mathrm{L}_S^{\mathrm{prop}}$, or to be of the form $t_1 + t_2$ or $t_1 \cdot t_2$ with S-terms t_1 and t_2. An *atomic S-formula* now is defined as an expression of the form $t_1 \leq t_2$ with S-terms t_1 and t_2. Finally, $\mathrm{L}_S^{\mathrm{pp}}$ consists of the closure of atomic S-formulas under boolean connectives.

An S-structure is a probability measure μ on the set W_S of truth assignments for S (more precisely: μ is a measure on the algebra $\mathscr{P}(W_S)$). To define the satisfaction relation $\models$ between μ and $\mathrm{L}_S^{\mathrm{pp}}$-formulas, first, for $\phi \in \mathrm{L}_S^{\mathrm{prop}}$ let

$$\mu(\phi) := \mu(\{w \in W_S \mid w(\phi) = \mathit{true}\}).$$

For an atomic S-formula $t_1 \leq t_2$ we then define $\mu \models t_1 \leq t_2$ iff by substituting for each probability expression $P(\phi)$ occurring in t_1 or t_2 the value $\mu(\phi)$ we obtain a valid numerical inequality. The obvious conditions for boolean connectives then complete the definition of the relation $\mu \models \phi$ for arbitrary $\phi \in \mathrm{L}_S^{\mathrm{pp}}$.

The relation $\models$ now provides a model theoretic presentation for an entailment relation $\vdash$ of $\mathscr{L}^{\mathrm{pp}}$.

Next, what Paris & Vencovská (1992) and Halpern & Koller (1995) have called an inference process (or procedure) is turned into a nonmonotonic entailment relation with a preferential model semantics for $\mathscr{L}^{\mathrm{pp}}$. We proceed by first giving a formal definition for a *measure selection function.*

Definition 5.3 For $n \geq 1$ let

$$\Delta^n := \{(x_1, \ldots, x_n) \in \mathbf{R}^n \mid x_i \geq 0, \textstyle\sum x_i = 1\}.$$

For each $n \geq 1$ let $I_n : \mathscr{P}(\Delta^n) \to \mathscr{P}(\Delta^n)$ with $I_n(G) \subseteq G$ for all $G \subseteq \Delta^n$, and $I_n(\pi(G)) = \pi(I_n(G))$ for all permutations π of $\{1, \ldots, n\}$. We then call $I := \cup_n I_n$ a measure selection function

While I is here defined to only work on subsets of Δ^n, it clearly induces a selection rule for measures on any finite algebra $\mathfrak{A}$: the set of probability measures on $\mathfrak{A}$, denoted $\Delta\mathfrak{A}$, after ordering the atoms $a_1, \ldots, a_n$ of $\mathfrak{A}$ in an arbitrary way, can be identified with Δ^n, and a measure selection function I can be used to select a subset $I(G)$ for every $G \subseteq \Delta\mathfrak{A}$. The condition of I_n being compatible with permutations makes $I(G)$ independent of the chosen ordering of the atoms of $\mathfrak{A}$.

Any measure selection function I now induces a preferential model semantics for $\mathscr{L}^{\mathrm{pp}}$. For $\Phi \subseteq \mathrm{L}_S^{\mathrm{pp}}$ we here denote $\mathrm{Mod}(\Phi, S) \subseteq \Delta\mathscr{P}(W_S)$ by $\Delta(\Phi, S)$, and define a partial order on $\Delta(\Phi, S)$ simply by $\mu < \mu'$ iff $\mu \in I(\Delta(\Phi, S))$ and $\mu' \notin I(\Delta(\Phi, S))$. Hence, in this ordering, $I(\Delta(\Phi, S)) = \lfloor \Delta(\Phi, S) \rfloor$. The nonmonotonic logic defined by adding the resulting nonmonotonic entailment relation to $\mathscr{L}^{\mathrm{pp}}$ is denoted by $\mathscr{L}_I^{\mathrm{pp}}$. As already noted in section 2, $\mathscr{L}_I^{\mathrm{pp}}$ may or may not have the reduct property for $\vdash$, depending on I.

In many cases the measure selection function I is defined by maximizing some (permutation invariant) function $K : \Delta^n \to \mathbf{R} \cup \{\infty\}$. In that case we actually have a global preferential model semantics defined by the preference relation on $\Delta\mathscr{P}(W_S)$: $\mu < \mu'$ iff $K(\mu) > K(\mu')$. The most prominent example, of course, being I_{me}, the maximum entropy selection rule. In some other interesting cases, however, we really only have a local preferential model semantics. The nonmonotonic logic $\mathscr{L}_{I_{\mathrm{cm}}}^{\mathrm{pp}}$ obtained from the center of mass selection rule I_{cm}, for example is not defined by a global preferential model semantics.

The definitions given so far now provide the formal framework for our introductory example from section 1: we get $\tau_1 \mathrel{|\!\sim} P(\mathit{LoM}) = 0.6$ and $\tau_2 \mathrel{|\!\sim} P(\mathit{ALoM} \vee \mathit{PLoM}) = 0.75$ in $\mathscr{L}_{I_{\mathrm{me}}}^{\mathrm{pp}}$. To make precise the representation dependence of maximum-entropy inference we first define a suitable subclass of $\mathit{Int}(\mathscr{L}^{\mathrm{pp}})$.

Let S, S' be finite propositional vocabularies, $f : \mathrm{L}_S^{\mathrm{prop}} \to \mathrm{L}_{S'}^{\mathrm{prop}}$ a propositional interpretation. In a straightforward way f then induces an abstract interpretation f^* of $\mathrm{L}_S^{\mathrm{pp}}$ into $\mathrm{L}_{S'}^{\mathrm{pp}}$ by inductively defining: $f^*(r) :\equiv r$ for rational constants r, $f^*(P(\phi)) :\equiv P(f(\phi))$ $(\phi \in \mathrm{L}_S^{\mathrm{prop}})$, and extending these definitions to arbitrary S-terms and $\mathrm{L}_S^{\mathrm{pp}}$-formulas via the canonical definitions for $+, \cdot, \leq, \neg$, and $\vee$.

Though this will be fairly obvious, we may formally appeal to lemma 3.4 in order to show that the mapping f^* really is an abstract interpretation with $\Omega(f^*) = \emptyset$: for $\mu \in \Delta\mathscr{P}(W_{S'})$ let $\bar{f}^*(\mu) \in \Delta(\mathscr{P}(W_S))$ be defined by

$$\bar{f}^*(\mu)(w) := \mu(\{w' \in W_{S'} \mid \bar{f}(w') = w\}) \quad (w \in W_S).$$

To see that (2) holds for μ and $\bar{f}^*(\mu)$ we may use an induction on the structure of $\phi \in \mathrm{L}_S^{\mathrm{pp}}$, founded on the following identity for $\gamma \in \mathrm{L}_S^{\mathrm{prop}}$:

$$\begin{aligned} \mu(f(\gamma)) &= \mu(\{w' \in W_{S'} \mid w'(f(\gamma)) = \mathit{true}\}) \\ &= \mu(\{w' \in W_{S'} \mid \bar{f}(w')(\gamma) = \mathit{true}\}) \\ &= \bar{f}^*(\mu)(\{w \in W_S \mid w(\gamma) = \mathit{true}\}) \\ &= \bar{f}^*(\mu)(\gamma). \end{aligned}$$

$\mathit{PI}(\mathscr{L}^{\mathrm{pp}})$ denotes the class of all abstract interpreta-

tions of $\mathscr{L}^{\mathrm{pp}}$ derived from propositional interpretations in the manner here described.

In the sequel, once again, we use f both for the underlying propositional interpretation, and the induced interpretation for $\mathscr{L}^{\mathrm{pp}}$. Note that then, whereas the argument of f may be either $\phi \in \mathrm{L}_S^{\mathrm{prop}}$ or $\phi \in \mathrm{L}_S^{\mathrm{pp}}$, the argument of the associated operator $\bar{f}$ can be either a truth assignment w, or a probability measure μ.

To continue the discussion of our example, let $S^* = \{A\}$, $f : A \mapsto LoM$, $g : A \mapsto ALoM \vee PLoM$. Then τ_1 and τ_2 are respectively $f(\alpha)$ and $g(\alpha)$ for $\alpha \equiv P(A) \geq 0.6$. Furthermore, $\tau_1 \overset{f,g}{\longleftrightarrow} \tau_2$, but for $\beta :\equiv P(A) = 0.6$: $\tau_1 \mid\!\sim f(\beta)$, $\tau_2 \not\mid\!\sim g(\beta)$. Thus, $\mathscr{L}^{\mathrm{pp}}_{I_{\mathrm{me}}}$ is not r.i. with respect to any $\boldsymbol{F}$ with $(f, g, \tau_1, \tau_2) \in \boldsymbol{F}$.

We now show how $\mathscr{L}^{\mathrm{pp}}_{I_{\mathrm{me}}}$ and any other nonmonotonic logic of the type $\mathscr{L}^{\mathrm{pp}}_I$ can be modified, so as to be r.i. with respect to

$$\boldsymbol{F}_P := \{(f, g, (\Phi, S_1), (\Psi, S_2)) \mid f, g \in PI(\mathscr{L}^{\mathrm{pp}}), \Phi \subseteq f(\mathrm{L}^{\mathrm{pp}}_{S^*}),\ \Psi \subseteq g(\mathrm{L}^{\mathrm{pp}}_{S^*})\},$$

which clearly contains the tuple (f, g, τ_1, τ_2) from above.

To this end, we first take a closer look at the set $\Delta(\Phi, S)$ for $\Phi \subseteq \mathrm{L}_S^{\mathrm{pp}}$. By definition, $\Delta(\Phi, S)$ is a set of probability measures on the algebra $\mathscr{P}(W_S)$. Now consider $\Delta(\tau_2, S_2)$: this is a set of probability measures on $W_{S_2} = \{w_1, \ldots, w_4\}$, where $S_2 = \{ALoM, PLoM\}$. $\Delta(\tau_2, S_2)$ is defined by a single constraint on the subset of truth assignments w_i for which $w_i(ALoM \vee PLoM) = \mathit{true}$. Assume that these are just w_1, w_2, w_3. Then, in order to decide whether $\mu \in \Delta(\tau_2, S_2)$, we only have to consider the restriction of μ to the subalgebra

$$\mathfrak{A}_{\Delta(\tau_2, S_2)} := \{\emptyset, \{w_1, w_2, w_3\}, \{w_4\}, W_{S_2}\}$$

of $\mathscr{P}(W_{S_2})$. Generally, for every finite algebra $\mathfrak{A}$, $G \subseteq \Delta\mathfrak{A}$, and subalgebra $\mathfrak{A}' \subseteq \mathfrak{A}$, we say that G *is defined by constraints on* $\mathfrak{A}'$ iff

$$\forall \mu :\ \mu \in G \ \text{ iff } \ \mu \restriction \mathfrak{A}' \in \{\nu \restriction \mathfrak{A}' \mid \nu \in G\} =: G \restriction \mathfrak{A}'.$$

In (Jaeger 1995) it is shown that when $G \subseteq \Delta\mathfrak{A}$ is defined by constraints on $\mathfrak{A}' \subseteq \mathfrak{A}$ and by constraints on $\mathfrak{A}'' \subseteq \mathfrak{A}$, then G is also defined by constraints on $\mathfrak{A}' \cap \mathfrak{A}''$. Thus, there always exists a unique smallest algebra $\mathfrak{A}_G \subseteq \mathfrak{A}$, such that G is defined by constraints on $\mathfrak{A}_G$. For any measure selection function I we may therefore define a modified measure selection function $\tilde{I}$ by

$$\tilde{I}(G) := \{\nu \mid \nu \restriction \mathfrak{A}_G \in I(G \restriction \mathfrak{A}_G)\}. \tag{6}$$

Returning to our example, we here obtain

$$\begin{aligned} \Delta(\tau_1, S_1) \restriction \mathfrak{A}_{\Delta(\tau_1, S_1)} &= \Delta(\tau_2, S_2) \restriction \mathfrak{A}_{\Delta(\tau_2, S_2)} \\ &= \{(x_1, x_2) \in \Delta^2 \mid x_1 \geq 0.6\}. \end{aligned}$$

(assuming suitable corresponding orderings of the atoms of $\mathfrak{A}_{\Delta(\tau_1, S_1)}$ and $\mathfrak{A}_{\Delta(\tau_2, S_2)}$). The maximum entropy element of this set is (0.6,0.4). Using the modified maximum entropy selection rule $\tilde{I}_{\mathrm{me}}$ we then get

$$\begin{aligned} \tilde{I}_{\mathrm{me}}(\Delta(\tau_1, S_1)) &= \{(0.6, 0.4)\} \\ \tilde{I}_{\mathrm{me}}(\Delta(\tau_2, S_2)) &= \{(x_1, x_2, x_3, x_4) \in \Delta^4 \mid x_1 + x_2 + x_3 = 0.6\}, \end{aligned}$$

and hence in $\mathscr{L}^{\mathrm{pp}}_{\tilde{I}_{\mathrm{me}}}$: $\tau_2 \mid\!\sim P(ALoM \vee PLoM) = 0.6$, as well as $\tau_1 \mid\!\sim P(LoM) = 0.6$.

A general result on the representation independence of logics $\mathscr{L}^{\mathrm{pp}}_{\tilde{I}}$ with respect to $\boldsymbol{F}_P$ can now be proven essentially by showing that the identity $\Delta(\Phi, S_1) \restriction \mathfrak{A}_{\Delta(\Phi, S_1)} = \Delta(\Psi, S_2) \restriction \mathfrak{A}_{\Delta(\Psi, S_2)}$ generally holds when $\Phi \overset{f,g}{\longleftrightarrow} \Psi$. However in certain cases it may only be true that $\Delta(\Phi, S_1) \restriction \mathfrak{A}_{\Delta(\Phi, S_1)} = \Delta(\Psi, S_2) \restriction \mathfrak{A}_{\Delta(\Psi, S_2)} \times \{0\}$, or $\Delta(\Psi, S_2) \restriction \mathfrak{A}_{\Delta(\Psi, S_2)} = \Delta(\Phi, S_1) \restriction \mathfrak{A}_{\Delta(\Phi, S_1)} \times \{0\}$, i.e. one of the two sets may consist of measures having an additional component, not corresponding to a component of measures from the other set, with constant value 0. In such cases we also write $G[\times 0] = J$ to express that either $G = J$ or $G \times \{0\} = J$. On account of this possibility we have to limit our general theorem to measure selection functions I that satisfy the condition: for all n, $G \subseteq \Delta^n$: $I(G \times \{0\}) = I(G) \times \{0\}$. We call such functions I *dimension independent*. Observe that all common selection functions are dimension independent.

Theorem 5.4 Let I be a dimension independent measure selection function, and $\tilde{I}$ defined by (6). Then $\mathscr{L}^{\mathrm{pp}}_{\tilde{I}}$ is r.i. with respect to $\boldsymbol{F}_P$.

Proof: Let $(f, g, (\Phi, S_1), (\Psi, S_2)) \in \boldsymbol{F}_P$, $\Phi \overset{f,g}{\longleftrightarrow} \Psi$. The theorem is proved using lemma 4.7 by showing that $\bar{f}(\tilde{I}(\Delta(\Phi, S_1))) = \bar{g}(\tilde{I}(\Delta(\Psi, S_2)))$, which is accomplished by proving the identities

$$\bar{f}(\tilde{I}(\Delta(\Phi, S_1))) = \tilde{I}(\bar{f}(\Delta(\Phi, S_1))) \tag{7}$$

(and the corresponding statement for $\bar{g}, \Psi$) and

$$\bar{f}(\Delta(\Phi, S_1)) = \bar{g}(\Delta(\Psi, S_2)). \tag{8}$$

Let $W_{S^*} = \{w_1, \ldots, w_n\}$. The (possibly empty) sets $\bar{f}^{-1}(w_i) \subseteq W_{S_1}$ $(i = 1, \ldots, n)$ form a partition of W_{S_1} and therefore are the atoms of a subalgebra $\mathfrak{A}_f$ of $\mathscr{P}(W_{S_1})$. A subalgebra $\mathfrak{A}_{\bar{f}}$ of $\mathscr{P}(W_{S^*})$

is defined by joining all the $w_i \in W_{S^*}$ for which $\bar{f}^{-1}(w_i) = \emptyset$, i.e. the atoms of $\mathfrak{A}_{\bar{f}}$ are the sets $\{w_i\}$ with $\bar{f}^{-1}(w_i) \neq \emptyset$ and $A_{\bar{f}} := \{w_i \mid \bar{f}^{-1}(w_i) = \emptyset\}$ (if this set is nonempty). We can view $\bar{f} : \bar{f}^{-1}(w_i) \mapsto \{w_i\}$ as an embedding of $\mathfrak{A}_f$ into $\mathfrak{A}_{\bar{f}}$, which is an isomorphism iff $A_{\bar{f}} = \emptyset$. Furthermore, taking measures on $\mathfrak{A}_{\bar{f}}$ and $\mathfrak{A}_f$ to be tuples in $\Delta^{k(+1)}$ defined by the corresponding orderings $(w_{i_1}, \ldots, w_{i_k}, A_{\bar{f}})$ and $(\bar{f}^{-1}(w_{i_1}), \ldots, \bar{f}^{-1}(w_{i_k}))$ $(w_{i_j} \in W_{S^*}, \bar{f}^{-1}(w_{i_j}) \neq \emptyset)$ we obtain

$$\Delta(\Phi, S_1) \restriction \mathfrak{A}_f[\times 0] = \bar{f}(\Delta(\Phi, S_1)) \restriction \mathfrak{A}_{\bar{f}}. \qquad (9)$$

From the condition that $\Phi \subseteq f(\mathrm{L}^{\mathrm{pp}}_{S^*})$ it follows that $\Delta(\Phi, S_1)$ is defined by constraints on $\mathfrak{A}_f$, because the evaluation of any probability term $P(f(\alpha))$ $(\alpha \in \mathrm{L}^{\mathrm{prop}}_{S^*})$ that may occur in some $\phi \in \Phi$ only depends on the measure of the set $\cup\{\bar{f}^{-1}(w_i) \mid w_i \in W_{S^*}, w_i(\alpha) = \mathit{true}\} \in \mathfrak{A}_f$. Also, $\bar{f}(\Delta(\Phi, S_1))$ is defined by constraints on $\mathfrak{A}_{\bar{f}}$, because $\bar{f}(\mu)(w_i) = 0$ for all $\mu \in \Delta(\Phi, S_1)$ and $w_i \in A_{\bar{f}}$. Therefore, from (9) we obtain

$$\begin{aligned}\Delta(\Phi, S_1) \restriction \mathfrak{A}_{\Delta(\Phi,S_1)}[\times 0] \\ = \bar{f}(\Delta(\Phi, S_1)) \restriction \mathfrak{A}_{\bar{f}(\Delta(\Phi,S_1))}, \qquad (10)\end{aligned}$$

where corresponding atoms of $\mathfrak{A}_{\bar{f}(\Delta(\Phi,S_1))}$ and $\mathfrak{A}_{\Delta(\Phi,S_1)}$ now are of the form $\{w_{i_1}, \ldots, w_{i_l}\}$ and $\{\bar{f}^{-1}(w_{i_1}), \ldots, \bar{f}^{-1}(w_{i_l})\}$ $(w_{i_j} \in W_{S^*})$. In case that $\mu(\bar{f}^{-1}(w_{i_j})) = 0$ for some $w_{i_j} \in W_{S^*}$ with $\bar{f}^{-1}(w_{i_j}) \neq \emptyset$ and all $\mu \in \Delta(\Phi, S_1)$ there also is a pair of corresponding atoms of the form $\{w_{i_1}, \ldots, w_{i_l}, A_{\bar{f}}\}$ and $\{\bar{f}^{-1}(w_{i_1}), \ldots, \bar{f}^{-1}(w_{i_l})\}$, so that $\mathfrak{A}_{\bar{f}(\Delta(\Phi,S_1))}$ and $\mathfrak{A}_{\Delta(\Phi,S_1)}$ are actually isomorphic.

By the dimension independence of I it follows from (9) and (10) that

$$\tilde{I}(\Delta(\Phi, S_1) \restriction \mathfrak{A}_f)[\times 0] = \tilde{I}(\bar{f}(\Delta(\Phi, S_1)) \restriction \mathfrak{A}_{\bar{f}}) \qquad (11)$$

from which (7) now follows by $\bar{f}(\tilde{I}(\Delta(\Phi, S_1))) = \bar{f}(\tilde{I}(\Delta(\Phi, S_1)) \restriction \mathfrak{A}_f))$, $\tilde{I}(\bar{f}(\Delta(\Phi, S_1))) = \tilde{I}(\bar{f}(\Delta(\Phi, S_1)) \restriction \mathfrak{A}_{\bar{f}})$, and the fact that for $\mu \in \Delta\mathfrak{A}_f$, $\mu' \in \Delta\mathfrak{A}_{\bar{f}}$ with $\mu[\times 0] = \mu'$ we have $\bar{f}(\mu) = \mu'$.

To prove (8) let $\nu \in \Delta(\Psi, S_2)$. For each $w_i \in W_{S^*}$ let ω_i be a propositional S^*-formula that is true exactly under the truth assignment w_i. For $w_i \in A_{\bar{f}}$ we now have $\vdash P(f(\omega_i)) = 0$, because $f(\omega_i)$ is unsatisfiable. From $\Phi \overset{f,g}{\longleftrightarrow} \Psi$ it then follows that $\Psi \vdash P(g(\omega_i)) = 0$, and hence $\bar{g}(\nu)(\{w_i\}) = 0$. It follows that $\mu(\bar{f}^{-1}(w_i)) := \bar{g}(\nu)(\{w_i\})$ defines a probability measure on $\mathfrak{A}_f$. Taking any extension of μ to $\mathcal{P}(W_{S_1})$ we receive an S_1-structure μ with $\bar{f}(\mu) = \bar{g}(\nu)$.

To see that $\mu \in \Delta(\Phi, S_1)$ let $\phi \in \Phi$, $\phi \equiv f(\alpha)$ for some $\alpha \in \mathrm{L}^{\mathrm{pp}}_{S^*}$. Then $\mu \models \phi$ iff $\bar{f}(\mu) \models \alpha$ iff $\bar{g}(\nu) \models \alpha$ iff $\nu \models g(\alpha)$. By $\Phi \overset{f,g}{\longleftrightarrow} \Psi$ and $\nu \models \Psi$ this last condition clearly is satisfied, and the inclusion from left to right of (8) is proven. The other inclusion follows from the symmetrical argument. □

The theorem is no longer true when the condition $\Phi \subseteq f(\mathrm{L}^{\mathrm{pp}}_{S^*})$, $\Psi \subseteq g(\mathrm{L}^{\mathrm{pp}}_{S^*})$ is dropped from the definition of $\boldsymbol{F}_P$: for $\tau_3 :\equiv P(ALoM \vee PLoM) \geq 0.6 \wedge P(ALoM) < 0.4$ we still have $\tau_1 \overset{f,g}{\longleftrightarrow} \tau_3$. But now $\Delta(\tau_3, S_2)$ is defined by constraints on no smaller algebra than $\mathcal{P}(W_{S_2})$ itself, and $\tilde{I}_{\mathrm{me}}(\Delta(\tau_3, S_2)) = I_{\mathrm{me}}(\Delta(\tau_3, S_2)) = \emptyset$. Thus, we get $\tau_3 \vdash \mathit{false}$, yet $\tau_1 \nvdash \mathit{false}$ in $\mathscr{L}^{\mathrm{pp}}_{\tilde{I}_{\mathrm{me}}}$.

$\boldsymbol{F}_P$ clearly contains the sets $\boldsymbol{F}_{RP}$, $\boldsymbol{F}_R$ and $\boldsymbol{F}_{LLE}$ of examples 4.2-4.4. Thus, for any dimension independent measure selection function I, $\tilde{I}$ possesses the reduct-, renaming-, and left logical equivalence property. For example, $\tilde{I}_{\mathrm{cm}}$ now has the reduct property that I_{cm} is lacking.

The modified maximum entropy selection rule, $\tilde{I}_{\mathrm{me}}$ can be shown to retain the property of I_{me} that Halpern and Koller (1995) have called "enforcing minimal irrelevance": when $\Phi \cup \{\phi\} \subseteq \mathrm{L}^{\mathrm{pp}}_S$, $\Psi \subseteq \mathrm{L}^{\mathrm{pp}}_{S'}$ with $S \cap S' = \emptyset$, then $\Phi \vdash \phi$ iff $\Phi \cup \Psi \vdash \phi$ in $\mathscr{L}^{\mathrm{pp}}_{\tilde{I}_{\mathrm{me}}}$. Clearly, $\mathscr{L}^{\mathrm{pp}}_{\tilde{I}_{\mathrm{me}}}$ also is not trivial in the sense of being "essentially entailment" (Halpern & Koller 1995). Finally, the representation independence of $\mathscr{L}^{\mathrm{pp}}_{\tilde{I}_{\mathrm{me}}}$ with respect to $\boldsymbol{F}_P$ implies that the measure selection rule $\tilde{I}_{\mathrm{me}}$ is representation independent (for state spaces consisting of propositional truth assignments and knowledge bases in $KB(\mathscr{L}^{\mathrm{pp}})$) in Halpern and Koller's sense. This result puts Theorem 3.10 in (Halpern & Koller 1995) somewhat into perspective: it shows that this theorem must actually rely on Halpern and Koller's condition that a measure selection function I to qualify as an inference procedure must preserve consistency, i.e. $I(G) \neq \emptyset$ for every $G \neq \emptyset$. This rather severe condition is not satisfied by I_{me}, $\tilde{I}_{\mathrm{me}}$, or most other natural selection rules. Therefore, these rules are outside the scope of theorem 3.10 as stated in (Halpern & Koller 1995). The selection function $\tilde{I}_{\mathrm{me}}$ shows that this theorem can not be extended to selection rules that may violate the preservation of consistency condition.

6 CONCLUSION

We have developed a very general and purely logical framework for analyzing representation independence of nonmonotonic logics. Representation independence does not emerge as a property as clear cut and un-

equivocal as, say, compactness. Rather, it comes (or fails to come) in many different forms and strengths, in our approach expressed by the parameter F. Two specific (classes of) logics have here been studied in some detail: rational closure logic, which has been found to exhibit a degree of representation independence more or less typical for qualitative nonmonotonic logics based on propositional logic, and propositional probabilistic logics, some important instances of which are notorious for their strong representation dependence. We have also seen that these latter logics can be modified so as to make them more representation independent, while at the same time retaining at least some of the desirable properties of the original inference rules.

The specific applications of the general theory given in this paper were rather easy by being restricted to a propositional background and correspondingly simple classes of abstract interpretations. When one moves to logics based on first-order predicate logic things become more complicated, as the concept of an interpretation becomes much more powerful. For this reason one will not be able to obtain results as clean as theorem 5.2 for circumscription, for example.

References

Boutelier, C. (1994), 'Conditional logics of normality: a modal approach', *Artificial Intelligence* **68**, 87–154.

Clark, K. L. (1978), Negation as failure, *in* H. Gallaire & J. Minker, eds, 'Logic and Databases', Plenum Press, New York.

Diaconis, P. & Zabell, S. (1982), 'Updating subjective probability', *Journal of the American Statistical Association* **77**(380), 822–830.

Ebbinghaus, H. D. (1985), Extended logics: The general framework, *in* J. Barwise & S. Feferman, eds, 'Model-Theoretic Logics', Springer-Verlag, pp. 25–76.

Frisch, A. & Haddaway, P. (1994), 'Anytime deduction for probabilistic logic', *Artificial Intelligence* **69**, 93–122.

Gabbay, D. (1985), Theoretical foundations for nonmonotonic reasoning in expert systems, *in* K. Apt, ed., 'Logics and Models of Cuncurrent Systems', Springer-Verlag, Berlin.

Grove, A., Halpern, J. & Koller, D. (1992), Random worlds and maximum entropy, *in* 'Proc. 7th IEEE Symp. on Logic in Computer Science'.

Halpern, J. Y. & Koller, D. (1995), Representation dependence in probabilistic inference, *in* 'Proceedings of IJCAI-95', pp. 1853–1860.

Hodges, W. (1993), *Model Theory*, Cambridge University Press.

Jaeger, M. (1995), Default Reasoning about Probabilities, PhD thesis, Universität des Saarlandes.

Kraus, S., Lehmann, D. & Magidor, M. (1990), 'Nonmonotonic reasoning, preferential models and cumulative logics', *Artificial Intelligence* **44**, 167–207.

Lehmann, D. (1989), What does a conditional knowledge base entail?, *in* 'Proceedings First International Conference on Principles of Knowledge Representation and Reasoning'.

Lehmann, D. & Magidor, M. (1992), 'What does a conditional knowledge base entail?', *Artificial Intelligence* **55**, 1–60.

Makinson, D. (1994), General patterns in nonmonotonic reasoning, *in* 'Handbook of Logic in Artificial Intelligence and Logic Programming', Oxford University Press, pp. 35–110.

Nilsson, N. (1986), 'Probabilistic logic', *Artificial Intelligence* **28**, 71–88.

Paris, J. & Vencovská, A. (1990), 'A note on the inevitability of maximum entropy', *International Journal of Approximate Reasoning* **4**, 183–223.

Paris, J. & Vencovská, A. (1992), 'A method for updating that justifies minimum cross entropy', *International Journal of Approximate Reasoning* **7**, 1–18.

Pearl, J. (1990), System Z: A natural ordering of defaults with tractable applications to default reasoning, *in* 'Proceedings of TARK-90', pp. 121–135.

Reiter, R. (1980), 'A logic for default reasoning', *Artificial Intelligence* **13**, 81–132.

Shoham, Y. (1987), Nonmonotonic logics: Meaning and utility, *in* 'Proceedings of IJCAI-87'.

Spohn, W. (1990), A general non-probabilistic theory of inductive reasoning, *in* R. Shachter, T. Levitt, L. Kanal & J. Lemmer, eds, 'Uncertainty in Artificial Intelligence 4', Elsevier Science Publishers.

Nonmonotonic Reasoning

Value Minimization in Circumscription

Chitta Baral **Alfredo Gabaldon**
Department of Computer Science
University of Texas at El Paso
El Paso, TX 79968 U.S.A.
{*chitta,alfredo*}*@cs.utep.edu*

Alessandro Provetti
C.I.R.F.I.D.
Università di Bologna
Via Galliera 3, I-40121 Bologna, Italy
provetti@cirfid.unibo.it

Abstract

Minimization in circumscription has focussed on minimizing the extent of a set of predicates (with or without priorities among them), or of a formula. Although most circumscription formalisms allow varying of functions and other constants, no formalism to the best of our knowledge minimized functions. In this paper we introduce and motivate the notion of *value minimizing* a function in circumscription. In value minimizing we do not minimize the extent of the function; rather we minimize the value of the function. We show how Lifschitz's nested abnormality theories can be used to do value minimization.

1 Introduction

In this paper we deal with the circumscription of terms in default theories for knowledge representation. While circumscription of predicates minimizes their extent, we describe circumscription of terms as minimizing the interpretation of terms w.r.t. a given *weight* of objects in the domain.

To the best of our knowledge minimization in circumscription has focussed on minimizing the extent of a set of predicates (with or without priorities among them) [Grosof, 1991, Lifschitz, 1993], or of a formula [McCarthy, 1986]. Also, although minimization of functions is not ruled out, we did not find examples that dealt with minimizing functions. Most circumscription formalisms do allow varying of functions and other constants while circumscribing though.

To visualize our goal, consider a database of employees containing the relation

$$Employee(emp\_id, emp\_name, salary)$$

The logical counterpart of this database is a first-order axiomatization with Closed World Assumption, e.g. with circumscription of the predicate *Employee*. That is, the *extent* of the predicate *Employee* is minimal compatible with the constraints.

Now suppose we would like to add additional information that the employer tries his best to minimize the salary of each of his employees. To formalize this we need axioms that will allow us to prefer models of the theory where employees get paid lesser. For example, let us assume that the employees in our database are *John* and *Mike* with *100* and *200* as their corresponding *emp_id*. Now suppose our theory has two models:

$$M_1 = \{ Employee(100, John, 30000), Employee(200, Mike, 40000)\}$$

$$M_2 = \{ Employee(100, John, 35000), Employee(200, Mike, 43000)\}$$

The employer will of course prefer $\mathcal{M}_1$ over $\mathcal{M}_2$. The goal of this paper is to formalize the minimization used in the above example where $\mathcal{M}_1$ is minimal among the set of models $\{\mathcal{M}_1, \mathcal{M}_2\}$.

The formalization of the above example includes the constraint that an employee has a unique salary and a unique name. Hence, we can intuitively view the above minimization as minimizing a function F_{sal} that maps employee ids into salaries. Notice, however, that we are not minimizing the extent of the function[1], rather we are minimizing the value of the function.

Recently in [Baral et al., 1996a], we have discussed the relationship between specifications in the action description language $\mathcal{L}$ [Baral et al., 1996b] and circumscriptive theories. There we were faced with the minimization of a particular term, $sit\_map(S_N)$, that mapped the current-situation symbol S_N onto a sequence of actions, understood as the history of the

[1] To stay close to the usual circumscription terminology, we use "extent of a function" to refer to the domain of a function.

domain. This was required to formalize the assumption

"no actions occurred except those needed to explain the facts in the theory"

which is present in the semantics of $\mathcal{L}$.

The main focus of [Baral et al., 1996a] was in showing the equivalence of specifications in $\mathcal{L}$ and their axiomatization based on Nested Abnormality Theories, a novel circumscription schema proposed by Lifschitz [Lifschitz, 1995]. Minimizing the term $Sit\_map(S_N)$ was part of this axiomatization. We believe function minimization and the minimization of ground terms (term minimization) are interesting in their own right and need to be discussed explicitly and independently. This will allow other researchers to easily use them in their applications using circumscription.

With that in mind, in this paper we generalize and expand on our observations in [Baral et al., 1996a] and formalize the notion of minimizing functions[2] where the minimization is done not with respect to the extent of the function, but with respect to the value of the function. The minimization is relative to an ordering $\mathcal{R}$ over the elements of the universe. To distinguish it from the standard minimization in circumscription, we refer to this as *value minimization.*

Our discussion is carried out in the framework of Nested Abnormality Theories, and we believe it represents further evidence of the suitability of this circumscription schema for knowledge representation.

To start with, we use the translation of specifications in $\mathcal{L}$ to circumscriptive theories as a motivational example of the necessity of value minimization.

2 Motivating example: narratives

The motivations and the theoretical framework of our approach to actions are discussed in detail in [Baral et al., 1996b], which is a continuation of the proposal in [Gelfond and Lifschitz, 1992] for *provably correct theories of actions.* Let us just point out that we have a language $\mathcal{L}$ for describing a sequence of actions and observations, regarding the evolution of a domain. Unlike the basic situation calculus, $\mathcal{L}$ allows observations about actual happenings and values of fluents in actual situations (i.e., allows narratives). Unlike the basic event calculus [Kowalski and Sergot, 1986], $\mathcal{L}$ allows reasoning about actual and hypothetical situations, possibly combined, e.g. to represent "adaptive plans." The entailment of $\mathcal{L}$ theories is nonmonotonic and thus apt for reasoning with incomplete information. Moreover, new observations are assimilated by simple insertion of new facts into the theory.

[2]Our notion of minimization is also applicable to terms.

Recently, [Baral et al., 1996a] we have defined a translation from narrative description specifications in $\mathcal{L}$ into Nested Abnormality Theories [Lifschitz, 1995][3]. In the rest of this section we will focus on this translation, which we describe informally as made up of:

- a subtheory describing effects of actions performed in a domain which is described by means of boolean fluents;
- the axiomatization of *common-sense inertia*;
- observations, i.e. facts describing fluent-values at the initial situation as well as at intermediate stages of the evolution of the domain; and
- facts describing which actions have been previously performed, and their ordering in time.

These axioms are completed with an assumption[4] specifying that

> models of the theory are those that satisfy all the observations and —other things being equal— entail that a minimal number of actions have occurred.

The above assumption, which is truly a model preference criterion, has been captured in our NAT formalization in the following way.

Let us introduce some notation. Let sequences of actions be represented by terms like

$$A_n \ldots \circ A_2 \circ A_1 \circ \epsilon$$

read as "A_1 then A_2 then ... A_n" (constant ϵ represents the empty sequence). Next, we have a set of constants $S_0, S_1 \ldots$ for denoting states of affairs the domain has passed through. The function Sit_map maps each situation constant into a sequence of actions which, intuitively, describes what has happened up to the situation itself. Of course the situations are totally ordered along a time line, although the ordering itself can vary from model to model. However, the special constant symbol S_N is always associated to the *late-most* situation, so that $Sit\_map(S_N)$ is interpreted on the longest sequence, i.e. the complete *history* of the domain. For instance:

$$\mathcal{M} \models Sit\_map(S_0) = \epsilon$$
$$\mathcal{M} \models Sit\_map(S_1) = A_1 \circ \epsilon$$
$$\ldots$$
$$\mathcal{M} \models Sit\_map(S_N) = A_k \circ A_{k-1} \circ \ldots \epsilon$$

[3]See Appendix A for a brief introduction to Nested Abnormality Theories.

[4]This assumption is not present elsewhere, in approaches that either entail no extra actions, e.g. [Shanahan, 1995] and [Kakas and Miller, 1996], or make no assumptions at all about action occurrences, e.g. [Lin and Reiter, 1995].

Now it should be clear how minimizing $Sit\_map(S_N)$, i.e. mapping it on the shortest possible sequence of actions, captures the assumption discussed above. Here is a sketch of how we formalized this notion with NATs. First of all, assume the following two NATs, which define predicates $Prefix\_eq$ and $Subsequence$ with their usual meaning (universal quantification is implicit on all variables):

$B_{Subsequence} =$
$\{min\ Subsequence\ :$
$\quad Subsequence(\epsilon, \epsilon)$
$\quad Subsequence(\alpha, \alpha_1) \supset Subsequence(\alpha, a \circ \alpha_1)$
$\quad Subsequence(\alpha, \alpha_1) \supset Subsequence(a \circ \alpha, a \circ \alpha_1)$
$\}$

$B_{prefix\_eq} =$
$\{min\ Prefix\_eq\ :$
$\quad Sequence(\alpha) \supset Prefix\_eq(\alpha, \alpha)$
$\quad Prefix\_eq(\alpha, \alpha_1) \supset Prefix\_eq(\alpha, a \circ \alpha_1)$
$\}$

The above two NATs are *nested* within the outermost theory $T_{\mathcal{L}}$, defined as follows.

$T_{\mathcal{L}} = \{\ Sit\_map\ :$

$\quad Sit\_map(S_0) = \epsilon$
$\quad Prefix\_eq(Sit\_map(s), Sit\_map(S_N))$

$(\star)\ Subsequence(\alpha, Sit\_map(S_N)) \supset Ab(\alpha)$

$\quad B_{prefix\_eq}$

$\quad B_{subsequence}$

$\quad$ other axioms
$\}$

The first two axioms make sure that S_0 is always mapped on the empty sequence and that all situations are mapped on sequences which are prefixes of the complete history $Sit\_map(S_N)$. Axiom $(\star)$ postulates that each model of $T_{\mathcal{L}}$ contains a set of instances $Ab(\alpha)$, where αs are all the possible subsequences of $Sit\_map(S_N)$. By including axiom $(\star)$ we exploit the fact that subsequence is a partial order on sequences, i.e. if α is a subsequence of β then all subsequences of the former are also subsequences of the latter.

Now, suppose interpretation $\mathcal{I}$ satisfies all the axioms of $T_{\mathcal{L}}$ and maps $Sit\_map(S_N)$ on α while interpretation $\mathcal{I}'$ maps it —other things being equal— onto β. As a result, the extent of Ab under $\mathcal{I}$ is a proper subset of the extent of Ab under $\mathcal{I}'$. Therefore, $\mathcal{I}'$ is not a circumscriptive model of $T_{\mathcal{L}}$. This kind of considerations are the subject of next section.

3 Generalizing the Approach

In this section the theory of value minimization by NATs is introduced and discussed in model-theoretic terms. First of all, the concept of value minimization is defined precisely. Second, we describe the structure of NATs that *implements* value minimization and show its correctness.

Our discussion is carried out in a theoretical framework where several assumptions are present. We argue that these assumptions are common to most applications of knowledge representation and reasoning we are familiar with, and therefore they do not seem to be limiting the scope of application.

Domain closure (DCA). Each object of the material domain is represented by a ground term, i.e., we restrict to models whose universe is isomorphic to the Herbrand domain.

Unique names (UNA). The usual sets of constant inequalities $C_1 \neq C_2$, $C_1 \neq C_3, \ldots$ are always included in the theories we examine.

Notice how both DCA and UNA are easily added or removed from theories. See [Lifschitz, 1995] for a discussion on which axioms/blocks are needed to implement DCA.

Definition 1 *(Explicit domain)*

A NAT is said to have an explicit domain if it contains axioms and blocks for DCA and UNA.

Now we can define value minimization as a partial order on models of explicit domain theories. To do so, let us introduce the following notation. Let

$$\mathcal{I}[[\pi]]$$

stand for the set of tuples which belong to the extent of predicate π in interpretation $\mathcal{I}$. For functions we use $\mathcal{I}[[\phi]]$ to denote the codomain of ϕ in $\mathcal{I}$. Also, we use

$$\mathcal{I}[[\phi]](\tau)$$

to denote the object which —according to interpretation $\mathcal{I}$— function ϕ maps term τ into. Since we are considering Herbrand models only, we can have expressions like $\mathcal{I}[[\phi]](\tau) = \nu$ where ν is a term of the language of the object theory.

3.1 Value Minimization

Definition 2 *(Value-minimal)*

Let T be a theory, ϕ be a function and Z a tuple of predicate/function constants in the language of T. Let $\mathcal{R}$ be a partial order defined between the elements of the universe. For two structures $\mathcal{M}$ and $\mathcal{M}'$ of T, we say $\mathcal{M} \leq^{(\phi,\mathcal{R});Z} \mathcal{M}'$ if

1. $|\mathcal{M}| = |\mathcal{M}'|$;
2. $\mathcal{M}[[\sigma]] = \mathcal{M}'[[\sigma]]$
   For each constant σ s.t. $\sigma \neq \phi$, $\sigma \notin Z$;
3. $\forall x. \mathcal{M}[[\phi]](x)\ \mathcal{R}\ \mathcal{M}'[[\phi]](x)$.

A model $\mathcal{M}$ of T is minimal relative to $\leq^{(\phi,\mathcal{R});Z}$ if there is no model $\mathcal{M}'$ of T such that $\mathcal{M}' <^{(\phi,\mathcal{R});Z} \mathcal{M}$.

The definition above can be tailored to minimization of ground terms, as we did in Section 2 when we minimized the ground term $Sit\_map(S_N)$ rather than function constant Sit_map as such. In these cases we speak of *term minimization*. In general, when the minimization is relative to a function ϕ and terms $t_1, \ldots, t_n$ the third condition in Definition 2 becomes

$$\mathcal{M}[[\phi]](t_i)\ \mathcal{R}\ \mathcal{M}'[[\phi]](t_i) \text{ for each } t_i,$$

and we write minimality as relative to the ordering $\leq^{(\phi,t_1,\ldots,t_n,\mathcal{R});Z}$ between structures.

Example 1 *(Exponentiation) Consider the following two interpretations of function power on the domain of naturals with $\leq$ as the usual "less than or equal" relation on naturals. By abuse of language, we define:*

$$\mathcal{M}[[power]](x) = x^2,\ \mathcal{M}'[[power]](x) = x^3.$$

Of course, for all x $\mathcal{M}[[power]](x) \leq \mathcal{M}'[[power]](x)$ and $\mathcal{M} <^{(power,\leq)} \mathcal{M}'$

Definition 2 above concerns minimization relative to a fixed, external criterion $\mathcal{R}$, which is a partial ordering between elements of the universe.

An interesting feature of NATs is that $\mathcal{R}$ can actually be *implemented* within the theory, simply by adding a block containing its axiomatization. Other circumscription methods (such as prioritized circumscription) would bring about the additional complication of possible undesired interaction between the part of the theory where $\mathcal{R}$ is supposed to be defined and the rest of the theory where $\mathcal{R}$ may appear[5].

Definition 3 *(Term ordering)*

We say that a NAT T has a term ordering $\mathcal{R}$ with respect to a function ϕ if it contains a block defining a partial ordering $\mathcal{R}$ on the elements of the codomain of ϕ, i.e. For all models $\mathcal{M}$ of T, and for all x and y,

$$\mathcal{M}[[\phi]](x)\ \mathcal{R}\ \mathcal{M}[[\phi]](y) \text{ iff } (\mathcal{M}[[\phi]](x), \mathcal{M}[[\phi]](y)) \in \mathcal{M}[[\mathcal{R}]]$$

□

[5]Notice that there is no extra complexity involved in this process, since transitive closures are always defined as a second-order theory, as indeed the block defining $\mathcal{R}$ is.

Now, let us show that for each value minimization criterion $\leq^{(\phi,\mathcal{R});Z}$ an equivalent NAT formulation is found. This result is the counterpart of fundamental Proposition 2.5.1 of [Lifschitz, 1993].

Theorem 1 *(Value-minimal equivalence)*

Let T be an explicit domain NAT with term ordering $\mathcal{R}$ and let ϕ be a function constant and Z be a tuple of predicate/function constants in the language of T. Then, a model $\mathcal{M}$ of T is a model of

$$\{\phi, Z : \forall x, y.\ \mathcal{R}(y, \phi(x)) \supset Ab(x, y)\ ;\ T\} \tag{1}$$

if and only if $\mathcal{M}$ is minimal relative to $\leq^{(\phi,\mathcal{R});Z}$.

Proof:

Consider a theory T as described in the above proposition. Let $T_{\phi,\mathcal{R}}$ be the NAT of the form (1) built from T.

($\Leftarrow$)

Assume $\mathcal{M}$ to be a model of T minimal relative to $\leq^{(\phi,\mathcal{R});Z}$. Let us define the structure $\mathcal{M}_{Ab}$ from $\mathcal{M}$ by letting:

1. $|\mathcal{M}_{Ab}| = |\mathcal{M}|$;
2. $\mathcal{M}_{Ab}[[\sigma]] = \mathcal{M}[[\sigma]]$ for every constant σ in the language of T[6] which is not included in Z;
3. $\mathcal{M}_{Ab}[[Ab]] = \{(x, y) : (y, \phi(x)) \in \mathcal{M}[[\mathcal{R}]]\}$.

Since $\mathcal{M}_{Ab}$ is a model of T, $\mathcal{M}_{Ab}$ is a model of $T_{\phi,\mathcal{R}}$ iff the extent of Ab is minimal. We now show $\mathcal{M}_{Ab}$ is a model of $T_{\phi,\mathcal{R}}$ by contradiction.

Suppose there is a model $\mathcal{M}'$ of $T_{\phi,\mathcal{R}}$ such that $|\mathcal{M}'| = |\mathcal{M}_{Ab}|$, $\mathcal{M}'[[\sigma]] = \mathcal{M}_{Ab}[[\sigma]]$ for every constant σ different from Ab and not in Z, and $\mathcal{M}'[[Ab]] \subset \mathcal{M}_{Ab}[[Ab]]$. Then, there exist x, y s.t.

$$(x, y) \notin \mathcal{M}'[[Ab]] \text{ and } (x, y) \in \mathcal{M}_{Ab}[[Ab]].$$

For the same x and y, since

$$\mathcal{M}[[\phi]](x)\ \mathcal{R}\ \mathcal{M}'[[\phi]](x) \Rightarrow y\ \mathcal{R}\ \mathcal{M}'[[\phi]](x)$$

by transitivity, we conclude that $\mathcal{M} \not\leq^{(\phi,\mathcal{R});Z} \mathcal{M}'$. On the other hand, since for all x

$$\mathcal{M}'[[\phi]](x)\ \mathcal{R}\ \mathcal{M}'[[\phi]](x) \Rightarrow (x, \mathcal{M}'[[\phi]](x)) \in \mathcal{M}'[[Ab]]$$

[6]Notice Ab does not belong to the language of T.

we can conclude that $(x, \mathcal{M}'[[\phi]](x)) \in \mathcal{M}_{Ab}[[Ab]]$. Now, since it follows that $\forall x.\mathcal{M}'[[\phi]](x)\ \mathcal{R}\ \mathcal{M}[[\phi]](x)$ we have established that $\mathcal{M}' \leq^{(\phi,\mathcal{R});Z} \mathcal{M}$. This implies $\mathcal{M}$ is not minimal relative to $\leq^{(\phi,\mathcal{R});Z}$ which contradicts our assumption, therefore $\mathcal{M}_{Ab}$ is a model of $T_{\phi,\mathcal{R}}$.

($\Rightarrow$)

Assume $\mathcal{M}$ to be a model of $T_{\phi,\mathcal{R}}$. By definition of NATs, $\mathcal{M}$ is a model of T. We show $\mathcal{M}$ is a model of T minimal relative to $\leq^{(\phi,\mathcal{R});Z}$ by contradiction.

Suppose there is a model $\mathcal{M}'$ of T s.t. $\mathcal{M}' <^{(\phi,\mathcal{R});Z} \mathcal{M}$. Let us augment $\mathcal{M}'$ by a predicate Ab defined as follows:

$$\mathcal{M}'[[Ab]] = \{(x, y) : (y, \phi(x)) \in \mathcal{M}'[[\mathcal{R}]]\}$$

Then, for all x $\mathcal{M}'[[\phi]](x)\ \mathcal{R}\ \mathcal{M}[[\phi]](x)$. By transitivity, for all x, y:

$$y\ \mathcal{R}\ \mathcal{M}'[[\phi]](x) \Rightarrow y\ \mathcal{R}\ \mathcal{M}[[\phi]](x)$$

Thus, $(x, y) \in \mathcal{M}'[[Ab]]$ implies $(x, y) \in \mathcal{M}[[Ab]]$ and therefore

$$\mathcal{M}'[[Ab]] \subseteq \mathcal{M}[[Ab]].$$

On the other hand, there exists an x for which the condition $\mathcal{M}[[\phi]](x)\ \mathcal{R}\ \mathcal{M}'[[\phi]](x)$ does not hold. For such an x, $(x, \mathcal{M}[[\phi]](x)) \notin \mathcal{M}'[[Ab]]$. However, for all x, $\mathcal{M}[[\phi]](x)\ \mathcal{R}\ \mathcal{M}[[\phi]](x)$ holds by definition of $\mathcal{R}$. Hence, $(x, \mathcal{M}[[\phi]](x)) \in \mathcal{M}[[Ab]]$ and we obtain

$$\mathcal{M}[[Ab]] \not\subseteq \mathcal{M}'[[Ab]]$$

Consequently, $\mathcal{M}$ is not a model of $T_{\phi,\mathcal{R}}$, which contradicts our assumption. Therefore $\mathcal{M}$ is a model of T minimal relative to $\leq^{(\phi,\mathcal{R});Z}$. □

3.2 Term Minimality

Let us now move on to define minimization of ground terms, i.e. to generalize the approach we discussed in Section 2. As before, consider an explicit domain NAT T with term ordering $\mathcal{R}$, and let ϕ be a function constant, $t_1, \ldots, t_n$ be ground terms, and Z be a tuple of predicate/function constants in the language of T. The following result is easily established.

Proposition 1 *(Term-minimal equivalence)*

a model $\mathcal{M}$ of T is a model[7] *of*

$$\{\phi, Z : \forall y.\ \mathcal{R}(y, \phi(t_1)) \supset Ab(t_1, y) \quad \ldots \quad \forall y.\ \mathcal{R}(y, \phi(t_n)) \supset Ab(t_n, y) \quad T \} \tag{2}$$

if and only if $\mathcal{M}$ is minimal relative to $\leq^{(\phi,t_1,\ldots,t_n,\mathcal{R});Z}$.

□

Our motivating example in Section 2 is a case of term-minimization. The following corollary of the above proposition is useful in proving the equivalence of the semantics of a domain description in $\mathcal{L}$ and its translation to the NAT $T_{\mathcal{L}}$.

Corollary 1 *Let T be the theory consisting of all the axioms in $T_{\mathcal{L}}$ from Section 2 except for axiom $Subsequence(\alpha, Sit\_map(S_N)) \supset Ab(\alpha)$. Then M is a model of $T_{\mathcal{L}}$ iff M is a model of T and is minimal relative to $\leq^{(Sit\_map, S_N, Subsequence)}$.*

□

4 Relationship With Other Circumscription Schemata

As pointed out earlier, one of the main advantages of using NATs is ease of representation, since the problem of interference between circumscription of different predicates is not present. Although NATs constitute a new circumscription framework, it is a predicate which is circumscribed in each block. Thus, intuitively, one may conclude that it should be possible to cast value minimization of a function into a more traditional circumscription form. Proposition 2 below gives a characterization of value minimization in a form similar to formula circumscription [McCarthy, 1986]. The difference with formula circumscription is that a function, rather than predicates, varies during the circumscription of the formula. In this sense, *this formulation constitutes a generalization of formula circumscription.*

Following the approach in [McCarthy, 1986], let $T_{\mathcal{R}}(\Phi, z)$ be a second order theory where function variable Φ[8] and predicate/function variables in z appear as free variables, and which contains a term ordering $\mathcal{R}$. Intuitively, Φ is the function whose valued is to be minimized with respect to $\mathcal{R}$ and z is the list of predicate/function variables which are allowed to vary during circumscription. Now, for a function constant ϕ and constants Z ($\mathcal{R} \notin Z$) in the language of $T_{\mathcal{R}}$ we can prove the following.

[7] For $n = 1$ we just have $Ab(y)$ in the right hand side of the axiom.

[8] This definition can be extended to allow a tuple of functions.

Proposition 2 *A structure $\mathcal{M}$ is a model of*

$$T_{\mathcal{R}}(\phi, Z) \wedge \forall \phi', z.[T_{\mathcal{R}}(\phi', z) \wedge [\forall x, y.\mathcal{R}(y, \phi'(x)) \supset \mathcal{R}(y, \phi(x))] \supset [\forall x, y.\mathcal{R}(y, \phi'(x)) \equiv \mathcal{R}(y, \phi(x))]] \quad (3)$$

if and only if $\mathcal{M}$ is a model of $T_{\mathcal{R}}(\phi, Z)$ and it is minimal relative to $\leq^{(\phi,\mathcal{R});Z}$. □

Formula (3) above can be simplified to the following:

$$T_{\mathcal{R}}(\phi, Z) \wedge \forall \phi', z.[T_{\mathcal{R}}(\phi', z) \wedge [\forall x.\mathcal{R}(\phi'(x), \phi(x))] \supset [\forall x.\mathcal{R}(\phi'(x), \phi(x)) \wedge \mathcal{R}(\phi(x), \phi'(x))]]$$

and clearly this can be further simplified to the formula

$$T_{\mathcal{R}}(\phi, Z) \wedge \forall \phi', z.[T_{\mathcal{R}}(\phi', z) \wedge [\forall x.\mathcal{R}(\phi'(x), \phi(x))] \supset [\forall x.\mathcal{R}(\phi(x), \phi'(x))]]$$

When minimization is relative to ground terms $t_1, \ldots, t_k$, we just need to replace the inner universally quantified formulae with the following conjuncts:

$$T_{\mathcal{R}}(\phi, Z) \wedge \forall \phi', z.\left[T_{\mathcal{R}}(\phi', z) \wedge \left[\bigwedge_{i=1,\ldots,k} \mathcal{R}(\phi'(t_i), \phi(t_i))\right] \supset \left[\bigwedge_{i=1,\ldots,k} \mathcal{R}(\phi(t_i), \phi'(t_i))\right]\right]$$

Furthermore, given that we only consider theories where objects in the world are represented by ground terms, there is an even simpler formula for term minimization when the list Z is empty. Let $\bar{c}$ stand for a tuple of ground terms $c_1, \ldots, c_k$, not containing the constant ϕ, where k is the number of terms relative to which we want to minimize ϕ. Let $T_{\mathcal{R}}(\bar{c})$ stand for $T_{\mathcal{R}} \bigwedge_{i=1,\ldots,k} \phi(t_i) = c_i$. Then we can write term minimization as:

$$T(\bar{c}) \wedge \left[\forall \bar{c}' \left(T(\bar{c}') \bigwedge_{i=1,\ldots,k} \mathcal{R}(c_i', c_i)\right) \supset \left(\bigwedge_{i=1,\ldots,k} \mathcal{R}(c_i, c_i')\right)\right]$$

Consider again our motivating example from Section(2). In this case, the list Z is empty, and we minimize function *Sit_map* only w.r.t. term S_N. Let T be the theory consisting of all the axioms in $T_{\mathcal{L}}$ except for axiom $Subsequence(\alpha, Sit\_map(S_N)) \supset Ab(\alpha)$. Let α be a sequence of actions and let us shorten *Subsequence* into *Sseq*. Let $T(\alpha)$ denote $T \wedge (Sit\_map(S_N) = \alpha))$. We can prove the following:

Proposition 3 *NAT $T_{\mathcal{L}}$ is equivalent to the formula*

$$T(\alpha) \wedge \forall \beta.[(T(\beta) \wedge Sseq(\beta, \alpha)) \supset Sseq(\alpha, \beta)]$$

□

5 Additional Remarks

We now make some observations about our formulation of value minimization of functions (and terms) using circumscription and possible extensions of it.

1. NATs' feature of allowing the definition of $\mathcal{R}$ to be in an independent block which is not affected by other axioms in the theory is very important for our purpose.
2. The blocks implementing value-minimization of a function are just another kind of block, and therefore can be part of another NAT. It was only for the sake of definition that we described them as outermost w.r.t. theories.
3. The concepts of minimizing the value of functions and terms presented so far can be extended to predicates, particularly when they intuitively encode functions. Consider again the database example where we have a predicate *Employee* with attributes *emp_id*, *emp_name*, and *emp_salary*. We want to minimize the salary of each employee. This can be done by the following NAT[9]:

   $\{Employee :$
   $Employee(x, n, z) \wedge y \leq z \wedge Salary(y) \wedge Salary(z) \supset Ab(x, y)$
   T
   $\}$

   where T consists of the definitions for $\leq$ and *Salary* and the rest of the theory about the employee database.

   This case arises in databases, where it is referred to as a *functional dependency* from a subset of the attributes of the predicate to another subset of the attributes.
4. The ordering $\leq^{(\phi,\mathcal{R});Z}$ between structures in Definition 2 can be extended to value minimization of a set of functions and to incorporate possible priorities between them by following the standard approach.
5. Finally, let us point out how Lifschitz's NAT notation is readily adapted to our definitions. We will write

   $$\{C_1, \ldots, C_m,\ min_{\mathcal{R}}\ \phi : A_1, \ldots, A_n, A_{\mathcal{R}}\} \quad (4)$$

   —where $A_{\mathcal{R}}$ defines[10] $\mathcal{R}$— to denote blocks of the form

   $\{C_1, \ldots, C_m, \phi : \mathcal{R}(y, \phi(x)) \supset Ab(x, y), A_1, \ldots, A_n, A_{\mathcal{R}}\}$

   Intuitively, block (4) refers to a theory consisting of blocks $A_1, \ldots, A_n$ and the block $A_{\mathcal{R}}$ defining

[9] Both the schema below and the ordering between structures in Definition 2 can be easily generalized to arbitrary predicates.

[10] We are assuming that $A_{\mathcal{R}}$ is defined using an NAT block such that statements about $\mathcal{R}$ outside of $A_{\mathcal{R}}$ do not affect the definition of $\mathcal{R}$.

$\mathcal{R}$, and where value minimization of function ϕ is performed while predicate/function constants $C_1, \ldots, C_m$ are varying.

Acknowledgements

This work benefited from several discussions with Michael Gelfond and Vladimir Lifschitz.

Support was provided by the National Science Foundation under grant Nr. IRI-9211662 and IRI-9501577.

References

[Baral et al., 1996a] Baral C., Gabaldon A., and Provetti A., 1996. Formalizing Narratives using Nested Circumscription. In Proc. of *14th AAAI National Conference on Artificial Intelligence*, pages 652–657.

Also in the Working notes of the third symposium *Logical Formalizations of Commonsense Reasoning*, pages 15–24.

[Baral et al., 1996b] Baral C., Gelfond M., and Provetti A., 1996. Representing Actions: Laws, Observations and Hypothesis. To appear on *Journal of Logic Programming*, special issue on actions. Also available via *http://cs.utep.edu/chitta/chitta.html*

An earlier version appeared in the Working Notes of AAAI Spring Symposium *Extending Theories of Actions: Formal Theories and Practical Applications.*

[Gelfond and Lifschitz, 1992] Gelfond M. and Lifschitz V., 1992. Representing actions in extended logic programs. In *Joint International Conference and Symposium on Logic Programming*, pages 559–573.

[Grosof, 1991] Grosof B., 1991. Generalizing Prioritization. In Proc. of *KR - Principles of Knowledge Representation and Reasoning*, pages 289–300. Morgan Kauffman.

[Kakas and Miller, 1996] Kakas A. and Miller R., 1996. A simple Declarative Language for Describing Narratives With Actions. To appear on *Journal of Logic Programming*, special issue on actions.

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

[Lifschitz, 1993] Lifschitz V., 1993. Circumscription. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, pages 297-352. Oxford Un. Press.

[Lifschitz, 1995] Lifschitz V., 1995. Nested Abnormality Theories. *Artificial Intelligence*, 74:351-365.

[Lin and Reiter, 1995] Lin F. and Reiter R., 1995. How to Progress a Database II: the STRIPS Connection. *Proc. the of 14th IJCAI Conference*, pages 2001-2007.

[McCarthy, 1986] McCarthy J., 1986. Applications of circumscription to formalizing common sense knowledge. *Artificial Intelligence*, 26(3):89–116.

[McCarthy and Hayes, 1969] McCarthy J. and Hayes P., 1986. Some philosophical problems from the standpoint of artificial intelligence. In Meltzer B. and Michie D., editors, *Machine Intelligence*, 4:463–502. Edinburgh University Press.

[Miller and Shanahan, 1994] Miller R. and Shanahan M., 1994. Narratives in the Situation Calculus. *Journal of Logic and Computation*, 4(5):513–530.

[Pinto and Reiter, 1993] Pinto J. and Reiter R., 1993. Temporal reasoning in logic programming: A case for the situation calculus. In *Proc. of 10th International Conference in Logic Programming*, pages 203–221.

[Shanahan, 1995] Shanahan M., 1995. A Circumscriptive Calculus of Events. *Artificial Intelligence*, 77:249–284.

A Overview of Nested Circumscription

This abstract has been included for referee convenience.

Nested Abnormality Theories (NATs) are a novel circumscription [McCarthy, 1986, Lifschitz, 1993] technique introduced in [Lifschitz, 1995]. NATs allow the use of several abnormality predicates to specify a body of common-sense knowledge without their circumscription conflicting. Lifschitz introduces the notion of *block:* a set of axioms $A_1, \ldots, A_n$—possibly containing the abnormality predicate Ab— "describes" a set of predicates/constants $C_1, \ldots, C_m$, which are allowed to vary during circumscription of Ab. The notation for such a theory is

$$C_1, \ldots, C_m \ : \ A_1, \ldots, A_n. \tag{5}$$

where each A_i may itself be a block of form (5). A block may be viewed as a set of axioms describing a collection of predicates/constants, possibly depending on other descriptions in embedded blocks. Block (5) can be expressed in terms of the circumscription operator as follows:

$$\mathrm{CIRC}[A_1 \wedge \ldots \wedge A_n; Ab; C_1, \ldots, C_m]. \tag{6}$$

In addition to NATs, Lifschitz [Lifschitz, 1995] introduces the idea of replacing the predicate *"Ab"* by an existentially quantified variable. Noticing that Ab plays an auxiliary role and that the interesting consequences of the theory are those which *do not* contain Ab, Ab is replaced by an existentially quantified variable. To put it differently, if $F(Ab)$ denotes (6), and *"ab"* is a predicate variable of the same arity as Ab, we are interested in the consequences of the sentence $\exists ab F(ab)$. The effect of this modification is that abnormality predicates become local to the block where each of them is used.

A.1 Syntax and semantics of NATs

The following definitions are from [Lifschitz, 1995]. Let L be a second order language which does not include Ab. For every natural number k, let L_k be the language obtained by adding the k-ary predicate constant Ab to L. $\{C_1, \ldots, C_m \ : \ A_1, \ldots, A_n\}$ is a *block* if each $C_1, \ldots, C_m$ is a predicate or a function constant of L, and each $A_1, \ldots, A_n$ is a formula of L_k or a block.

A *Nested Abnormality Theory* is a set of blocks. The semantics of NATs is characterized by a mapping φ from blocks into sentences of L. If A is a formula of language L_k, φA stands for the universal closure of A, otherwise

$$\varphi\{C_1, \ldots, C_m \ : \ A_1, \ldots, A_n\} \ = \ \exists ab F(ab)$$

where

$$F(Ab) = \mathrm{CIRC}[\varphi A_1 \wedge \ldots \wedge \varphi A_n; Ab; C_1, \ldots, C_m]$$

For any NAT T, φT stands for $\{\varphi A | A \in T\}$. A *model* of T is a model of φT in the sense of classical logic. A *consequence* of T is a sentence ϕ of language L that is true in all models of T. In this paper, as suggested in [Lifschitz, 1993], we will use the abbreviation

$$\{C_1, \ldots, C_m, \ \mathit{min} \ P : A_1, \ldots, A_n\}$$

to denote blocks of the form

$$\{C_1, \ldots, C_m, P : P(x) \supset Ab(x), A_1, \ldots, A_n\}$$

As the notation suggests, this type of block is used when it is necessary to circumscribe a particular predicate P in a block. In [Lifschitz, 1993] it is shown that

$$\varphi\{C_1, \ldots, C_m, \ \mathit{min} \ P : A_1, \ldots, A_n\}$$

is equivalent to the formula

$$\mathrm{CIRC}[A_1 \wedge \ldots \wedge A_n; P; C_1, \ldots, C_m]$$

when each of A_i is a sentence.

For the sake of our investigation, we are specially interested in NATs for which an equivalent first-order formalization can be given. The idea is to *simplify* a theory of nested blocks by substituting the innermost blocks with their equivalent first-order theories, and repeat the process inside out. The final theory —if it exists— can be used for deduction or comparison purposes. Even when a first-order equivalent exists, the NAT is more suggestive and compact.

Biconsequence Relations for Nonmonotonic Reasoning

Alexander Bochman

e-mail: bochman@macs.cs.biu.ac.il

Abstract

In this study we suggest a general formalism for nonmonotonic reasoning. The formalism, called *biconsequence relations*, provides a general framework of reasoning with respect to a pair of contexts. As is shown in [Bochman 1996], it allows to give a uniform representation for various semantics for logic programs involving negation as failure. Here we will consider a generalization of the formalism obtained by incorporating classical inference. This generalization turns out to be sufficiently expressive to provide a representation for such systems of nonmonotonic reasoning as various default and modal nonmonotonic logics. It allows also to account for some recent attempts to extend semantics suggested for logic programs to general nonmonotonic formalisms.

Keywords: Representation formalisms, nonmonotonic logics.

Introduction

In this study we suggest a general formalism for nonmonotonic reasoning, called *biconsequence relations*. At its structural level, the formalism is based on a four-valued inference under Belnap's interpretation of the four truth-values [Belnap 1977]. The latter allows the propositions to be not only true or false, but also undetermined (neither true nor false) or contradictory (both true and false). This interpretation has been widely used in the literature in order to give a representation of reasoning in presence of incomplete and inconsistent information (see, e.g., [Levesque 1984, Fitting 1989, Sakama and Inoue 1995]). Moreover, it is shown in [Bochman 1996] that this formalism allows to give a uniform representation for various semantics of general logic programs involving negation as failure. What we are going to show here is that there is a fairly general way of 'lifting' this formalism to languages based on classical logic. As we will see, this extension of the basic formalism, called *classical biconsequence relations*, turns out to be sufficiently expressive to give a representation for such systems of nonmonotonic reasoning as various default logics and modal nonmonotonic logics. Moreover, due to its intimate connection with logic programming, it allows also to account for some recent attempts to extend semantics suggested for logic programs to other nonmonotonic formalisms (see, e.g., [Baral and Subrahmanian 1992, Lifschitz 1994, Przymusinska and Przymusinski 1994]).

1 Biconsequence Relations - An Overview

The logical formalism we suggest for representing nonmonotonic reasoning is based on a four-valued entailment. More exactly, we use an interpretation of the four truth-values suggested in [Belnap 1977] that amounts to their identification with all the (four) subsets of the set of classical truth-values $\{t, f\}$: a proposition can be not only true ($\{t\}$) or false ($\{f\}$), but also contradictory (both true and false, $\{t, f\}$) or undetermined (neither true nor false, $\emptyset$). This representation provides a natural connection between four-valued reasoning and 'real' problems of commonsense reasoning and thereby the main reason for using it in studying the latter.

The above interpretation allows us to identify any four-valued interpretation ν with a pair of *independent* classical valuations corresponding, respectively, to assignments of truth and falsity to propositions:

$$\nu \models A \equiv t \in \nu(A) \qquad \nu \dashv A \equiv f \in \nu(A)$$

Consequently, four-valued reasoning in general can still be seen as reasoning about truth and falsity of propositions, provided that the assignments of truth and falsity are independent of each other. This leads to the following construction that gives a syntactic formalization for such a reasoning.

By a *bisequent* we will mean a rule of the form $a : b \Vdash c : d$, where a, b, c, d are finite sets of propositions. The intended interpretation of such rules is

"If all a's are true and all b's are false, then one of c's is true or one of d's is false".

Accordingly, propositions from a and b will be called, respectively, *positive* and *negative premises*, while those from c and d - *positive* and *negative conclusions*. The following definition provides a characterization of such bisequents in accordance with their intended interpretation.

Definition 1.1 *A* biconsequence relation *is a set of bisequents that is closed with respect to the following rules:*

(Reflexivity) $A : \Vdash A : \qquad : A \Vdash : A$

(Monotonicity)

$$\frac{a : b \Vdash c : d}{a' : b' \Vdash c' : d'}, \text{ if } a \subseteq a', b \subseteq b', c \subseteq c', d \subseteq d'.$$

(Cut)

$$\frac{a : b \Vdash A, c : d \quad A, a : b \Vdash c : d}{a : b \Vdash c : d}$$

$$\frac{a : b \Vdash c : A, d \quad a : A, b \Vdash c : d}{a : b \Vdash c : d}$$

A biconsequence relation can be seen as a 'doubled' version of an ordinary sequent calculus reflecting the independence of the truth and falsity assignments. As in the latter, we can extend the notion of a bisequent to arbitrary sets of propositions by accepting an appropriate *compactness requirement*:

(Compactness)

$$u : v \Vdash w : z \quad \text{iff} \quad a : b \Vdash c : d,$$

for some finite sets a, b, c, d such that $a \subseteq u$, $b \subseteq v$, $c \subseteq w$ and $d \subseteq z$.

In what follows, we will denote by $\overline{u}$ the complement of a set u. The following definition describes 'canonical models' of biconsequence relations.

Definition 1.2 *A pair of sets of propositions (u, v) is a* bitheory *of a biconsequence relation $\Vdash$ if*

$$u : \overline{v} \nVdash \overline{u} : v.$$

The above description reflects a kind of closure condition with respect to bisequents. It is equivalent to the following more explicit closure requirement:

> If $a : b \Vdash c : d$ and $a \subseteq u, b \subseteq \overline{v}$, then either $c \cap u \neq \emptyset$ or $d \cap \overline{v} \neq \emptyset$.

Notice that any bitheory (u, v) can be identified with a four-valued interpretation by treating u as the set of true propositions and v as the set of propositions that are not false. The following Representation Theorem can then be used to show that biconsequence relations are adequate for their intended four-valued interpretation. The theorem shows, in effect, that any biconsequence relation is generated by a set of four-valued interpretations.

Theorem 1.1 (Representation Theorem) *$a : b \Vdash c : d$ iff, for any bitheory (u, v), if $a \subseteq u$ and $b \subseteq \overline{v}$, then either $c \cap u \neq \emptyset$ or $d \cap \overline{v} \neq \emptyset$.*

It can be shown that any four-valued connective is expressible in our formalism via introduction and elimination rules as in ordinary sequent calculi, the only distinction being that we will have a pair of introduction rules and a pair of elimination rules corresponding to two premise sets and two conclusion sets, respectively. As in the case of a classical sequent calculus, any bisequent involving four-valued connectives will be reducible, ultimately, to a set of bisequents containing atomic propositions only.

Biconsequence relations and logic programs. On our approach, bisequents can serve as direct representations of logic programming rules of a most general kind, namely those involving disjunction and negation in their heads (see, e.g., [Inoue and Sakama 1994] for a use of such rules). Thus, a rule

$$\mathbf{not}\, D_1 \vee \ldots \vee \mathbf{not}\, D_k \vee C_1 \vee \ldots \vee C_l \leftarrow$$
$$\leftarrow A_1 \wedge \ldots A_m \wedge \mathbf{not}\, B_1 \wedge \cdots \wedge \mathbf{not}\, B_n$$

is representable by a bisequent

$$A_1, \ldots, A_m : B_1, \ldots, B_n \Vdash C_1, \ldots, C_l : D_1, \ldots, D_k.$$

If S is a set of bisequents corresponding to program clauses, then the least biconsequence relation containing S can be considered as a *logical theory* corresponding to a logic program. It turns out that under this representation, all the logical connectives occurring in program clauses (including **not**) can be seen as four-valued connectives (see [Bochman 1996] for details). In this way we are restoring a connection between Logic and Logic Programming that has been lost with the introduction of logic programs involving a so-called negation by failure. The notion of negation-as-failure refers on this interpretation not to the meaning of **not**, but rather to a special nonmonotonic mechanism of 'jumping to conclusions' that allows to obtain additional negative conclusions. An appropriate extension of this mechanism to general nonmonotonic formalisms studied in this paper will be presented in Section 4.

2 Classical Biconsequence Relations

As we said above, the semantics of bisequents involves a pair of independent interpretations assigning, respectively, truth and falsity to propositions. However, we are not compelled to stick to this understanding. Generally speaking, biconsequence relations can be seen as

describing reasoning with respect to *pairs of contexts*. Two main alternative representations of such a pair, appropriate for our study, are as follows:

- We can assume that the first context represents what is actually *true*, while the second - what is *assumed* (not) to hold. On this reading, a bisequent $a : b \Vdash c : d$ can be seen as a conditional inference rule $a \Vdash c$ that is "fired" only under assumption that propositions from d hold, while those from b do not. Under this understanding we will obtain a formal representation of *assumption-based reasoning*. We will use this reading below to provide a formal interpretation for credulous inference determined by alternative default logics.
- We can assign an *epistemic* reading to the second context, namely as representing what is *known or believed* to hold. An alternative, thoroughly epistemic interpretation could be that the first context represents what is known, while the second - what is believed to hold. Both alternatives lead to a formal representation of *modal nonmonotonic reasoning*.

Despite the difference between the above interpretations, however, we will see that in both cases the basic constructions and models remain the same. In fact, one of the aims of this study consists in persuading the reader that we can abstract from these differences in doing nonmonotonic reasoning.

The above 'extended' interpretations allows us also to extend the formalism of biconsequence relations to languages that include classical logical connectives. The only additional general requirement we will impose on our semantics is that both contexts involved in such 'bi-interpretations' should respect classical inference. In other words, our only requirement will be that we should be able to reason classically in each of them.

Let $\mathcal{L}_c$ denote a language containing classical connectives $\neg, \wedge, \vee, \supset$, and $\vdash$ the ordinary classical consequence relation with respect to these connectives.

Definition 2.1 *A* classical biconsequence relation *(CBR) is a biconsequence relation in the language $\mathcal{L}_c$ satisfying the following conditions:*

(Supraclassicality)

$$\frac{a \vdash A}{a : \Vdash A :} \qquad \frac{a \vdash A}{: A \Vdash : a}$$

The above conditions can be also expressed without the use of $\vdash$. For example, a CBR can be defined equivalently as a biconsequence relation satisfying the following two conditions:

1. $\Vdash A :$ and $: A \Vdash$, for any classical tautology;
2. $A, A \supset B : \Vdash B :$ and $: B \Vdash : A, A \supset B$, for any propositions A and B.

The most important consequence of Supraclassicality is a possibility of replacement of classically equivalent formulas in bisequents. In addition, it allows to replace positive premises and negative conclusions by their conjunctions, though disjunctive conclusions are not replaceable in this way by their classical disjunctions. In this way the formalism preserves the distinction between classical disjunctions and more strong, 'epistemic' disjunctions employed, for example, in a theory of disjunctive default rules, suggested in [Gelfond et al. 1991]. Thus, the following rules for the classical disjunction $\vee$ are in general invalid:

$$A \vee B : \Vdash A, B : \qquad : A, B \Vdash : A \vee B.$$

As a consequence, $A : \Vdash C :$ and $B : \Vdash C :$ do not necessarily imply $A \vee B : \Vdash C :$.

Speaking more generally, CBRs are only *supra*classical consequence relations. This means, in particular, that the deduction theorem is not valid for the classical implication – $a, A : \Vdash B :$ does not imply $a : \Vdash A \supset B :$. Similarly, the Contraposition rule

$$\frac{A : \Vdash B :}{\neg B : \Vdash \neg A :}$$

is also invalid in general. It must be noted, however, that this 'deviation' from the classical behavior should not be seen as a weakness of our formalism. In fact, it is these distinctions that give the formalism its expressive power and allow us to represent information that is impossible to capture using only classical formulae.

2.1 Additional Conditions

We will describe now a number of natural additional conditions that can be imposed on a CBR. The first such condition is obtained by strengthening Supraclassicality to the case of classically inconsistent sets of propositions.

Definition 2.2 *A CBR $\Vdash$ will be called* $\neg$-consistent *if it satisfies the following two rules:*

$$\frac{a \vdash}{a : \Vdash} \qquad \frac{a \vdash}{\Vdash : a}$$

The above rules can be shown to be equivalent to the following two conditions:

$$A, \neg A : \Vdash \qquad \Vdash : A, \neg A$$

These conditions exclude, in effect, classically inconsistent sets of propositions from consideration and thereby reflect a requirement that each of the two contexts of reasoning should be classically consistent.

The following stronger rule, called *global* $\neg$-*consistency*, requires our pairs of contexts to be always classically compatible.

$$\frac{a, b \vdash}{a : \Vdash : b}$$

In fact, the rule is reducible to the following simple condition

$$\neg A : \Vdash : A$$

saying that if $\neg A$ is true, then A should be false (in Belnap's sense of the words). The rule can be shown to be equivalent to the condition that if (u, v) is a bitheory, then $u \cup v$ is a classically consistent set.

A further strengthening of the notion of a CBR can be obtained by adding each of the following structural rules:

(Consistency) $\quad A : A \Vdash$

(Completeness) $\quad \Vdash A : A$

Consistency amounts to the condition that any bitheory (u, v) should be *consistent*, that is, $u \subseteq v$. This condition will be used extensively in what follows. Similarly, Completeness requires bitheories to be *complete*, that is, $v \subseteq u$. Note that if a biconsequence relation satisfies both Consistency and Completeness, it is reducible to an ordinary sequent calculus. On the other hand, the 'common content' of both these conditions is given by

(CC) $\quad A : A \Vdash B : B$

The latter condition requires any bitheory to be either consistent or complete. In fact, the resulting biconsequence relation can be seen as an *intersection* of a consistent and a complete biconsequence relation. As is shown in [Bochman 1995a], this condition turns out to be admissible with respect to the well-founded semantics of normal logic programs. Note also that, given $\neg$-consistency, CC implies global $\neg$-consistency.

2.2 Semantics

Supraclassicality can be shown to be equivalent to the condition that both components of any bitheory should be deductively closed sets. The method of semantic representation for such objects is actually well known (see [Kraus et al. 1990]; cf. also [Dix and Makinson 1992]) and consists in representing any deductively closed set by the associated set of worlds.

Definition 2.3 *A* model *of a CBR is a triple* $\mathcal{M} = \langle W, S, I \rangle$, *where* W *is a set of worlds,* $S \subseteq 2^W \times 2^W$ *and* $I : \mathcal{P} \rightarrow 2^W$ *is a function assigning each propositional atom of the language a subset of* W. *Any pair of sets* $\nu = (U, V)$ *from* S *will be called a* bistate *of the model.*

Thus, a model is essentially a relation on sets of worlds. The validity of propositional formulas in a model will be determined with respect to pairs (ν, i), where ν is a bistate and i is a world:

- $\nu, i \models p$ iff $i \in I(p)$, where p is a propositional atom;
- $\nu, i \models \neg A$ iff $\nu, i \not\models A$;
- $\nu, i \models A \wedge B$ iff $\nu, i \models A$ and $\nu, i \models B$.

As can be seen, the bistate parameter is actually excessive at this stage, since the value of each classical formula is entirely determined by the relevant world. We will need this more general validation function later when we will consider modal extensions of the language.

Now we can define a four-valued interpretation corresponding to a bistate using a pair of (truth and falsity) assignments defined as follows: If $\nu = (U, V)$ is a bistate, then

$$\nu \models\!\models A \text{ iff } \nu, i \models A, \text{ for every } i \in U;$$
$$\nu \models\!\!\dashv A \text{ iff } \nu, i \not\models A, \text{ for some } i \in V$$

A bisequent $a : b \Vdash c : d$ will be said to be *valid* in a model $\mathcal{M}$ if, for any bistate ν from S,[1]

> if $\nu \models\!\models A$, for any $A \in a$, and $\nu \models\!\!\dashv B$, for any $B \in b$, then either $\nu \models\!\models C$, for some $C \in c$, or $\nu \models\!\!\dashv D$, for some $D \in d$.

For a model $\mathcal{M}$, we will denote by $\Vdash_{\mathcal{M}}$ the set of all bisequents that are valid in $\mathcal{M}$. The following Completeness Theorem shows that the resulting semantics provides an adequate semantic characterization of classical biconsequence relations.

Theorem 2.1 (Completeness) $\Vdash$ *is a classical biconsequence relation iff there is a model* $\mathcal{M}$ *such that* $\Vdash = \Vdash_{\mathcal{M}}$.

Using this theorem, various strengthenings of the notion of a CBR, described above, can be characterized by imposing restrictions on possible semantic models. For example, a characterization of $\neg$-consistent CBRs can be obtained by restricting the set of possible bistates to those that involve only non-empty sets of worlds. Similarly, the semantic condition corresponding to global $\neg$-consistency is that, for any bistate (U, V), $U \cap V \neq \emptyset$. Consistency corresponds to the requirement that $V \subseteq U$, for any bistate (U, V), while Completeness amounts to the inverse inclusion $U \subseteq V$.

2.3 Expansions

In this section we will introduce perhaps the most important nonmonotonic models definable in the context of a CBR. As we will see, these objects, called expansions, can be seen as a proper generalization of Reiter's extensions, Moore's stable expansions, $\mathcal{S}$-expansions of modal nonmonotonic logics and a number of other nonmonotonic models known in the literature. Thus,

[1] Cf. the Representation Theorem above.

expansions[2] form the main ingredient of Lifschitz models in his MBNF (see below). In addition, they correspond to *preferred models* of the bimodal logic GK suggested in [Lin and Shoham 1992].

Definition 2.4 *1. A bitheory (u, v) of a CBR will be called* positively minimal *if there is no bitheory (u', v) such that $u' \subset u$.*

2. A set of propositions u will be called an expansion *of a CBR if (u, u) is a positively minimal bitheory.*

Semantically, expansions correspond to sets of propositions that are true with respect to a bistate of the form form (U, U) such that there is no larger set of worlds U' for which (U', U) is also a bistate[3].

Some basic results concerning expansions have been established in [Bochman 1995b], though in a somewhat different terminology. Thus, the following lemma, stated in [Bochman 1995b], gives an alternative description of expansions:

Lemma 2.2 *A set u is an expansion iff*

$$A \in u \quad \textit{if and only if} \quad : \overline{u} \Vdash A, \overline{u} : u$$

An important fact, established in [Bochman 1995b], is that there are two structural rules for a CBR that preserve expansions when imposed on it. These rules are Consistency (see above) and

$$\textit{(Semi-Factoring)} \qquad \frac{a : b \Vdash b : a}{: b \Vdash : a}$$

The rule Semi-Factoring amounts to the requirement that, for any bitheory (u, v), (v, v) is also a bitheory. Given Consistency, it can be shown to be equivalent to the following rule (called G-Coherence in [Bochman 1995b]):

$$\frac{A, a : b \Vdash : d}{a : b \Vdash : d, A}$$

Proposition 2.3 *If $\Vdash^c$ is a least CBR containing $\Vdash$ and closed with respect to Consistency and Semi-Factoring, then it has the same expansions as $\Vdash$.*

Now, it turns out that for CBRs satisfying Consistency, expansions can be characterized as follows:

Proposition 2.4 *u is an expansion if and only if*

$$u = \{A \mid : \overline{u} \Vdash A : u\}$$

[2]We called these objects L-expansions in [Bochman 1995b].

[3]Cf. the corresponding definition in [Lifschitz 1994].

Thus, accepting an assumption-based interpretation of our pair of contexts, expansion can be seen as a set of formulas that are provable on the basis of taking precisely them as a set of our assumptions. This proposition implies also an interesting fact that expansions are determined ultimately by bisequents having only singular formulas as their positive conclusions. This latter fact will allow us to establish an important connection between the notion of expansion and stable expansions of autoepistemic logic.

2.4 Default Logics

Default consequence relations introduced in [Bochman 1994] are in fact a special case of classical biconsequence relations obtained by restricting bisequents to that of the form $a : b \Vdash A :$, that is, to bisequents that do not involve multiple or negative conclusions. As is shown there, default consequence relations can be seen as a generalization of Reiter's default logic. We can recast now these results directly in terms of biconsequence relations as follows. To begin with, the following translation provides a representation of default rules in terms of bisequents:

$$tr_R : \qquad A : b/C \quad \Rightarrow \quad A : \neg b \Vdash C :$$

In addition, if $\Delta = (W, D)$ is a default theory, we replace each axiom $A \in W$ with a bisequent $\Vdash A :$. Finally, if $tr_R(\Delta)$ is the set of all bisequents corresponding to rules and axioms of Δ, then we will denote by $\Vdash^R_\Delta$ the least CBR containing $tr_R(\Delta)$. Then we will have

Proposition 2.5 *If Δ is a default theory, then its extensions coincide with expansions of $\Vdash^R_\Delta$.*

Disjunctive Defaults. An extension of the notion of a default rule was suggested in [Gelfond et al. 1991]. *Disjunctive default rules* were introduced there in order to give a more appropriate representation in cases where common formalizations in terms of ordinary, singular default rules produced unintuitive results (such as in Poole's broken arm example). It is easy to show that the resulting default theories are still representable in terms of classical biconsequence relations and the notion of an expansion, provided we identify disjunctive default rules with bisequents of the form $a : b \Vdash c :$. It is interesting to note, however, that so far as expansions serve as intended nonmonotonic models, there is no *conceptual* necessity in introducing default rules with disjunctive conclusions: by Proposition 2.4, any expansion is determined, ultimately, by a set of bisequents with singular conclusions that can be inferred in a given CBR. Nevertheless, there are some complexity considerations (see [Eiter and Gottlob 1995]) showing that though disjunctive bisequents are eliminable in principle, the corresponding transformation of disjunctive

default theories into ordinary ones should be computationally expensive.

We will show now that the use of negative conclusions in bisequents allows to define some alternative forms of default reasoning. We begin with the following notion of credulous inference in the context of a CBR:

Definition 2.5 *A proposition A will be said to be a* credulous consequence *of a set of propositions a if there is a pair of sets b, d such that : $b \not\Vdash d$ and $a : b \Vdash A : d$. If $a = \emptyset$, we will say that A is* credulously provable *in* $\Vdash$.

According to an 'assumption-based' interpretation of bisequents, this definition says that A is a credulous consequence of a if A can be inferred ('explained') from a on the basis of some consistent set of positive and negative assumptions.

It turns out that the following translation of default rules into bisequents is appropriate for capturing a credulous inference determined by Lukaszewicz notion of justified extension [Lukaszewicz 1988].

$$tr_L : \quad a : b/A \quad \Rightarrow \quad a : \neg b \Vdash A : A$$

Retaining the previous translation of the axioms of a default theory, we will define $\Vdash^L_\Delta$ as the least CBR containing $tr_L(\Delta)$.

Proposition 2.6 *A proposition A is credulously provable in $\Vdash^L_\Delta$ if and only if it belongs to some justified extension of Δ.*

As an immediate consequence of this result, we obtain that, under this translation, the notion of a default proof from [Lukaszewicz 1988] (see also [Froidevaux and Mengin 1994]) is captured by our notion of credulous inference.

As our next result, we will show that expansions of the corresponding CBR are also adequate for capturing Lukaszewicz notion of a default proof. The following Proposition shows, in particular, that A is credulously provable in $\Vdash^L_\Delta$ if and only if it belongs to some expansion of $\Vdash^L_\Delta$.

Proposition 2.7 *1. Any justified extension of Δ is an expansion of $\Vdash^L_\Delta$.*

2. Any expansion of $\Vdash^L_\Delta$ is included in some justified extension of Δ.

Remark. It is interesting to mention that a default theory can be seen in this case as a generalization of an *abductive program* from [Inoue and Sakama 1994]. The latter is defined as a pair $\langle P, \Gamma \rangle$, where P is a general extended disjunctive program (corresponding, in our terminology, to a CBR in the language $\mathcal{L}_c$ restricted to classical literals), while Γ is a set of *assumptions with preconditions* that can be seen as default rules of the form $a : b/A$. As is shown in [Inoue and Sakama 1994], any such conditional assumption is representable by a program clause corresponding, on our interpretation, to a bisequent $a : b \Vdash A : A$. Consequently, a default theory (W, D) under Lukaszewicz interpretation can be identified with an abductive theory $\langle W, D \rangle$.

Finally, we will consider a credulous provability generated by a constrained default logic (see [Schaub 1991, Froidevaux and Mengin 1994]). According to the latter, default rules should be seen as comitting to their justifications. In addition, they should be seminormal, that is, the conclusion should be always included among the justifications of the rule. The following translation gives a relevant interpretation of default rules in terms of bisequents:

$$tr_C : \quad a : b/A \quad \Rightarrow \quad a : \not\Vdash A : A, b$$

Recall that the rules of CBR imply that negative conclusions can be replaced by their conjunction. As a result, justifications of any default rule are replaceable, modulo this translation, by their conjunction, and any rule $a : b/A$ will be equivalent in this sense to a semi-normal rule $a : b, A/A$. Notice also that the only distinction between tr_C and tr_L amounts to their respective treatment of justifications. While the former requires that they should be assumed to hold, the latter views them as saying that their negations are assumed not to hold.

As before, we will denote by $\Vdash^C_\Delta$ the least CBR containing translations of all default rules and axioms of a default theory Δ.

Proposition 2.8 *A is credulously provable in $\Vdash^C_\Delta$ if and only if it belongs to some constrained extension of a default theory Δ.*

To conclude this section, note that the above two alternative representations of default rules are still insufficient to provide a representation for a *sceptical* inference determined by the corresponding default logics. To see this, notice that both Lukaszewicz and constrained default logic have a peculiar feature that 'trivial' default rules of the form $A : B/t$, where t is a tautology, may change extensions when added to a default theory (though our translations make them trivial in both cases). This means that such rules have a nontrivial informational content. In fact, in both cases it amounts, roughly, to something like "If A holds, take B as an additional assumption, provided it is consistent with other assumptions made". Such 'default assumption' rules are not expressible directly in our framework.

3 Modalization

We will describe now some definable modal extensions of CBRs. It turns out that in most cases the relevant modal operators will be eliminable in the context of a CBR. What will be gained, however, by introducing them is that any bisequent will be representable by some modal formula. It turns out that by introducing appropriate definable modal operators of this kind, we obtain some familiar modal formalisms of nonmonotonic reasoning.

3.1 MBNF

Classical biconsequence relations were introduced first in [Bochman 1995b] as a formalism providing a nonmodal counterpart of Lifschitz' Modal logic of Belief and Negation as Failure (MBNF). In this section we will show that, starting with classical biconsequence relations, the full modal theory of MBNF can be 'restored' by introducing additional operators that are *definable* on the semantic structures characterizing CBRs. In other words, our aim here is to show that MBNF can be seen as a straightforward *definitional extension* of classical biconsequence relations.

The following two semantic definitions provide a description of a pair of natural modal operators definable in the semantic framework of CBRs. If $\nu = (U, V)$ is a bistate, then

$$\nu, i \models \mathcal{L}A \quad \text{iff} \quad \nu, j \models A, \text{ for any } j \in U;$$
$$\nu, i \models \mathcal{B}A \quad \text{iff} \quad \nu, j \models A, \text{ for any } j \in V.$$

Lifschitz' original operators **B** and **not** from [Lifschitz 1994] can be identified, respectively, with $\mathcal{L}$ and $\neg\mathcal{B}$. Then it can be seen that the above conditions, taken together with the semantic conditions for the classical connectives $\{\wedge, \neg\}$, give an equivalent semantic description of MBNF-interpretations; the only distinction is that instead of truth of a proposition in a world i with respect to a bistate $\nu = (U, V)$, Lifschitz speaks about truth of a proposition with respect to a triple (i, U, V). However, the above definitions with respect to worlds serve in our construction the only purpose of defining truth or falsity of propositions with respect to bistates. As a consequence, what is actually captured in our system from the 'full' MBNF is its *modalized* fragment that deals with formulas in which any propositional atom occurs in the scope of some modal operator. Fortunately, this restriction does not reduce the range of current applications of MBNF (see [Bochman 1995b] for details).

The resulting system can be completely axiomatized. Below are the rules that characterize $\mathcal{B}$:

$$\Vdash \mathcal{B}A : A \qquad : \mathcal{B}A \Vdash : A$$
$$: A \Vdash \neg\mathcal{B}A : \qquad : A, \neg\mathcal{B}A \Vdash$$

Similar rules can be given for $\mathcal{L}$. Using this axiomatization, it can be shown that above operators are always *eliminable*: any bisequent in a CBR is reducible to a set of bisequents involving neither $\mathcal{L}$ nor $\mathcal{B}$. However, using these operators and classical connectives, any bisequent can be transformed into a provable formula. Thus, a bisequent

$$A_1, \ldots, A_k : B_1, \ldots, B_l \Vdash C_1, \ldots, C_m : D_1, \ldots, D_n$$

can be shown to be equivalent to a bisequent

$$(1) \quad \Vdash (\mathcal{L}A_1 \& \ldots \& \mathcal{L}A_k \& \mathbf{not}\, B_1 \& \ldots \& \mathbf{not}\, B_l) \supset$$
$$\supset (\mathcal{L}C_1 \vee \cdots \vee \mathcal{L}C_m \vee \mathbf{not}\, D_1 \vee \cdots \vee \mathbf{not}\, D_n) :$$

This allows us to represent our classical biconsequence relations as MBNF-theories.

As has been shown in [Bochman 1995b], what was termed a nonmonotonic model of MBNF in [Lifschitz 1994] is actually a pair consisting of an expansion and a world. For modalized MBNF theories the world component of such models becomes inessential, and models are reduced, in effect, to expansions. Moreover, all representations of other nonmonotonic formalisms in MBNF studied in the paper, such as autoepistemic logic and default logic, use actually only a modalized fragment of MBNF. As a consequence, the relevant nonmonotonic objects of these formalisms were represented, in effect, as expansions. Consequently, our results about default logic and autoepistemic logic (see below) can be seen as a (non-modal) reformulation of the corresponding results from [Lifschitz 1994] and [Lin and Shoham 1992]. In addition, this establishes the correspondence between expansions and *answer sets* of logic programs of a most general kind (see below; cf. also [Lifschitz and Woo 1992, Inoue and Sakama 1994]).

3.2 Modal Nonmonotonic Logics

Recall that the semantic requirement corresponding to the Consistency rule is that, for any bistate (U, V), $V \subseteq U$. In this case, our bicomponent models can be seen as a proper generalization of the *minimal model semantics* for modal nonmonotonic logics from [Schwarz 1992]. Accordingly, we will show now that classical biconsequence relations are sufficiently expressive to capture nonmonotonic reasoning determined by modal nonmonotonic logics.

Let $\mathbb{S}$ be any modal logic containing the necessitation rule. As is well known (see [Marek et al. 1993]), $\mathbb{S}$-expansions of a modal theory are stable sets, and the latter are uniquely determined by their objective (nonmodal) subsets. It turns out that for any modal theory we can construct a CBR in the language without modalities such that its expansions are precisely objective subsets of $\mathbb{S}$-expansions of this theory.

Proposition 3.1 *For any modal theory, there is a CBR such that its expansions are precisely objective kernels of* S*-expansions of the theory.*

One particular case of this correspondence is especially interesting. Assuming Consistency, let us modify the definition of the operator $\mathcal{L}$ above as follows:

$$\nu, i \vDash \mathcal{L}A \quad \text{iff} \quad \begin{cases} \nu, j \vDash A, \forall j \in U, & \text{if } i \in U \setminus V; \\ \nu, j \vDash A, \forall j \in V, & \text{if } i \in V; \end{cases}$$

The modified operator behaves as before with respect to positive contexts, but now $\mathcal{B}$ is *definable as* $\neg\mathcal{L}\neg\mathcal{L}$. In particular, any bisequent is representable now by a formula involving only one modal operator, $\mathcal{L}$. As can be seen, the resulting semantic framework corresponds to a (unimodal) BK-semantics from [Schwarz and Truszczyński 1994] (cf. also [Bochman 1995b]). As is shown there, it gives a semantic interpretation of an important nonmonotonic modal logic **S4F**.

3.3 Autoepistemic logic

Moore's autoepistemic logic can also be identified with a simple definitional extension of our formalism. In fact, the relevant modal operator is just $\mathcal{B}$[4].

For any autoepistemic theory T in the classical language augmented with the operator $\mathcal{B}$, we will denote by $\Vdash^m_T$ the least CBR *in the same language* containing all formulas from T. Then we have

Proposition 3.2 *If T is an autoepistemic theory, then stable expansions of T coincide with expansions of $\Vdash^m_T$.*

Now, as we mentioned earlier, the rules for $\mathcal{B}$ allow to eliminate all occurrences of $\mathcal{B}$ in bisequents. Moreover, any bisequent $\Vdash A :$ is reducible in this way to a set of bisequents of the form $: b \Vdash C : d$ that do not contain $\mathcal{B}$. Let $\Vdash^o_T$ be the least CBR *in the nonmodal language* containing all such bisequents corresponding to formulas from T.

Corollary 3.3 *Expansions of $\Vdash^o_T$ are exactly objective kernels of stable expansions of T.*

In other words, autoepistemic logic can be described entirely in a non-modal language. More exactly, it corresponds to CBRs determined by bisequents of the form $: b \Vdash A : d$. Now notice that by Proposition 2.4, it is precisely these bisequents that are needed to characterize expansions in general. Consequently, we have

Proposition 3.4 *For any CBR $\Vdash$ there is an autoepistemic theory T such that expansions of $\Vdash$ are exactly objective kernels of stable expansions of T.*

[4] Notice that it has all the properties of a **K45**-operator.

It is possible to give a formal characterization of classical biconsequence relations corresponding to autoepistemic theories, that is, of CBRs generated by bisequents of the form $: b \Vdash A : d$.

A CBR will be called *(positively) singular*, if it satisfies the rules

$$A, \neg A : \Vdash \qquad \text{and} \qquad \Vdash A, \neg A :$$

On the semantic side, the above rules amount to the requirement that the positive component of any bistate contains a single world. As can be seen, in this case the classical connectives satisfy all the rules and conditions for classical inference with respect to the positive premises and conclusions of bisequents. In particular, any set of positive conclusions is replaceable by their classical disjunction, and the deduction theorem becomes valid. As a result, any bisequent $a : b \Vdash c : d$ of a singular CBR becomes equivalent to a bisequent $: b \Vdash (\wedge a) \rightarrow (\vee c) : d$, where $\wedge a$ $(\vee a)$ denotes the conjunction (respectively, disjunction) of elements from a. This shows, in other words, that singular CBR is actually a logical theory of bisequents of the form $: b \Vdash A : d$, that is, of bisequents without positive premises and with singular (non-disjunctive) positive conclusions. As follows from the results stated earlier in this section, these are precisely CBRs representing autoepistemic theories.

As we have seen, we do not need the full expressive capabilities of a CBR to describe autoepistemic logic. In fact, for any biconsequence relation we can define the corresponding *autoepistemic default consequence relation* from [Bochman 1994] as follows:

$$a : b \Vdash^a A \quad \text{iff} \quad : b \Vdash A : a$$

Such a consequence relation is already sufficient to describe the 'objective content' of autoepistemic logic (see [Bochman1994, 1995b] for details).

4 General Circumscription Schema

As we mentioned in the Introduction, the basic formalism of biconsequence relations allows to give a uniform representation for various semantics of logic programs, suggested in the literature. In this section we will show how these semantics can be extended to classical biconsequence relations. We can only be brief here.

The general construction of nonmonotonic semantics for logic programs, suggested in [Bochman 1996], is based on two principles, Positive Minimality and Coherence. The *Positive Minimality principle* provides the main nonmonotonic mechanism of 'jumping to conclusions' in the framework of biconsequence relations. It says that only positively minimal bitheories (interpretations) should matter in determining the truth-values of propositions, where minimality is defined

with respect to propositional atoms belonging to positive components of bitheories[5]. In accordance with this, the *circumscription* of a biconsequence relation $\Vdash$ is defined as a biconsequence relation determined (in the sense of the Representation Theorem) by the set of all positively minimal bitheories of $\Vdash$.

The *Coherence principle* amounts to imposing the following two rules on a biconsequence relation:

$$\text{(Coherence)} \qquad \frac{\Vdash A :}{: A \Vdash} \qquad \frac{A : \Vdash}{\Vdash : A}$$

The rules say, respectively, that provably true propositions should be non-false (and hence classically true), while provably non-true propositions should be classically false. Generally speaking, the Coherence principle gives a mechanism of deriving *classical information* in the four-valued setting.

An important feature of the Coherence principle is that it crucially depends on the underlying language in which a biconsequence relation is formulated. In other words, the actual force of the above rules depends on what propositions are susceptible to coherence, or, more exactly, on what four-valued logical connectives belong to the language. As is shown in [Bochman 1996], different semantics for logic programs are obtainable by varying the set of allowable connectives.

It has turned out, however, that in all cases of interest the relevant coherence rules are expressible in terms of some *structural rules* that do not involve explicit occurrences of connectives. Such rules will allow us to give appropriate extensions of the corresponding semantics to classical biconsequence relations.

A *nonmonotonic completion* of a biconsequence relation $\Vdash$ is defined in [Bochman 1996] as a biconsequence relation obtained from $\Vdash$ by first circumscribing it and then adding the coherence rules to the circumscribed biconsequence relation. It turns out that such a completion contains all the information that can be nonmonotonically inferred from the source biconsequence relation. As we already said, different semantics for logic programs can be obtained by varying the language of formulas admissible for the coherence rules.

In order to extend the above schema to classical biconsequence relations, we only need to take into account the fact that our basic language is now not the language of propositional atoms, but rather the language $\mathcal{L}_c$ of classical propositions. Thus, a positively minimal bitheory is defined now in accordance with Definition 2.4 that reflects minimality with respect to *all classical formulas*. As before, the *circumscription* of a classical biconsequence relation $\Vdash$ will be defined as a CBR $\Vdash^c$ that is determined by positively minimal bimodels of $\Vdash$.

[5]Cf. Definition 2.4 above.

As to coherence, it turns out that the range of possibilities will be drastically reduced due to the presence of the classical connectives in our language. We will give below some basic cases of our schema, corresponding to existing attempts of extending semantics for logic programs to 'full' nonmonotonic formalisms.

The well-founded semantics. The weakest completion (that already involves conjunctive formulas) is obtained by adding the following coherence rules to the (previously) circumscribed biconsequence relation:

$$\frac{\Vdash a :}{: a \Vdash} \qquad \frac{A : \Vdash}{\Vdash : A}$$

Such a completion corresponds to an extension of the well-founded semantics for ordinary default theories suggested, e.g., in [Baral and Subrahmanian 1992, Przymusinska and Przymusinski 1994].
Unfortunately, though such a semantics always exists for consistent theories, it has a serious drawback of being too cautious: in all cases that admit mutually inconsistent alternatives, it 'retreats' to the set of monotonically provable propositions. This holds even in cases when a theory involves independent pieces of data, witness the following default theory (see [Przymusinska and Przymusinski 1994]): $W = \emptyset$, $D = \{: A/A, \neg A/\neg A, : B/C\}$. The well-founded semantics for this theory contains only tautologies, though there seems to be no reason why the last default rule is not 'fired' to obtain C.

At that point the requirement of $\neg$-consistency (see Definition 2.2) changes the situation: in the resulting completion C is already provable. The requirement seems to be plausible also in many other respects. However, it has its price. First, some default theories will have no consistent completion, e.g., a theory $W = \{\neg B\}$, $D = \{: A/B, : A/A\}$. Second, partial stable semantics is no longer equivalent to the well-founded one for ordinary default theories, as it was the case with normal logic programs, so we have to choose between them.

Finally, it is worth to note that the coherence principle

$$\frac{\Vdash \neg A :}{\Vdash : A}$$

used in [Pereira and Alferes 1992] in defining a well-founded semantics with explicit negation (WFSX), implies the second $\neg$-consistency rule $\Vdash : A, \neg A$. Moreover, both turn out to be equivalent in the presence of the coherence rule $\dfrac{\Vdash A :}{: A \Vdash}$. Thus, $\neg$-consistency accounts for the intuitions behind the WFSX.

Stationary completion. The coherence rules for the case of the stationary semantics as it is defined in [Przymusinski 1995] amount jointly to the following rule:

$$\frac{a : \Vdash b :}{: b \Vdash : a}$$

As is shown in [Bochman 1996], the corresponding completion characterizes also a semantics of stable classes suggested in [Baral and Subrahmanian 1992]. Hence, our construction can be seen as a natural generalization of both these semantics to classical biconsequence relations.

Invariant completion. The coherence rules characterizing the partial stable semantics from [Przymusinski 1991] amount to the following rule:

$$\text{(Invariance)} \qquad \frac{a : b \Vdash c : d}{d : c \Vdash b : a}$$

The rule reflects a semantic condition that if (U, V) is a bistate, then (V, U) is also a bistate.

Stable completion. Finally, the coherence rules for the case of the stable (answer set) semantics amount to adding the Consistency and Completeness rules to the circumscribed biconsequence relation. The semantic condition corresponding to these rules is that any bistate should have the form (U, U). As an immediate consequence of this fact, we obtain that the bitheories of the nonmonotonic completion in this case coincide with expansions of the source CBR. This means, in particular, that expansions can be seen as semantic objects characterizing classical reasoning with respect to the circumscribed biconsequence relation.

5 Conclusions and Further Issues

In this study we suggested a general formalism for nonmonotonic reasoning that has allowed us to give a uniform representation of rather diverse approaches in this field. As is shown, in most cases the diversity has turned out to be superficial: the majority of these formalisms use the same nonmonotonic principle of 'jumping to conclusions', and the only difference between them consists in underlying logical formalisms used to derive monotonic consequences. Moreover, in many cases the difference can even be reduced to that of the *language* in which these systems are formulated. It should be noted that this uniform description has been made possible in a large part due to the fact that the formalism of biconsequence relations provides a *clear separation between logical and nonmonotonic aspects of nonmonotonic reasoning*.

Another important benefit of our approach is that it establishes an intimate connection between nonmonotonic reasoning and logic programming. The relationship between these two fields has been described mainly through translations of logic programs into different nonmonotonic formalisms, such as default logic, circumscription or modal nonmonotonic logics. From the viewpoint of our framework, the connection between these two fields turns out to be even closer. In our approach, program clauses can be directly identified with rules (sequents) of our formalism, and in this way the reasoning about logic programs and their semantics can be seen as a (most simple) kind of nonmonotonic reasoning in general. Moreover, the formalism of biconsequence relations itself can be seen as a straightforward formalization of the *logic(s)* behind logic programs. In this way we are restoring a connection between Logic and Logic Programming that has been lost with the introduction of logic programs involving a so-called negation by failure.

Another important result of this study is that we do not really need modal operators to represent nonmonotonic reasoning. From a point of view of expressivity, the only advantage of using the latter is that a rule-based inference can be transformed into a more familiar formula-based one. But it has its price, namely the need to account for nested modalities.

In many respects, our approach is similar to a theory of cumulative consequence relations [Kraus et al. 1990]. Actually, the most glaring omission of this study is how our approach is related to preferential entailment, since the latter is also considered as a general theory of nonmonotonic inference. The main difference between the two is that our approach, just as all the approaches to nonmonotonic reasoning it subsumes, is *fully explicit* in the sense that it does not presuppose existence of an additional preference relation imposed on the semantic structure; all the necessary information is already embodied in the rules. The need to 'disclose' some external ordering on propositions in interpreting nonmonotonic inference, be it an ordering of 'normality' or 'epistemic entrenchment', turns out to be a difficult task when our aim is not to justify a few conclusions known in advance, e.g. why Tweety flies, but rather to infer (possibly unexpected) consequences from a given body of data. An attractive way out of this difficulty is to try to restore such an ordering from the rules themselves, as is done in an approach originated by Judea Pearl [Pearl 1990, Goldszmidt and Pearl 1991]. This idea can be extended to biconsequence relations and suggests a way for a mutual fertilization of the two main trends in nonmonotonic reasoning.

References

[Baral and Subrahmanian 1992] C. Baral and V.S. Subrahmanian (1992) Stable and extension class theory for logic programs and default logics. *J. of Automated Reasoning*, **8**: 345–366.

[Belnap 1977] N.D. Belnap (1977) A useful four-valued logic. In M. Dunn and G. Epstein (eds.) *Modern Uses of Multiple-Valued Logic*, D. Reidel, Dordrecht, pp. 8–41.

[Bochman 1994] A. Bochman (1994) On the relation between default and modal consequence relations.

In *Proc. 4th Int. Conf. on Principles of Knowledge Representation and Reasoning, KR'94*, Morgan Kaufmann, San Francisco, CA., 1994, pp. 63–74.

[Bochman 1995] A. Bochman (1995) Modal nonmonotonic logics demodalized. *Annals of Mathematics and Artificial Intelligence* **15**: 101–123.

[Bochman 1995a] A. Bochman (1995) Default consequence relations as a logical framework for logic programs. *Proc. Third International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR'95*, Lecture Notes in Artificial Intelligence, 928, pp. 245–258.

[Bochman 1995b] A. Bochman (1995) On bimodal nonmonotonic logics and their unimodal and nonmodal equivalents. *Proc. IJCAI'95*, pp. 1518–1524.

[Bochman 1996] A. Bochman (1996) On logics and semantics for general logic programs. (submitted)

[Dix and Makinson 1992] J. Dix and D. Makinson (1992) The relationship between KLM and MAK models for nonmonotonic inference operations. *J. of Logic and Computation* **1**: 131–140.

[Eiter and Gottlob 1995] T. Either and G. Gottlob. On the computational cost of disjunctive logic programming: propositional case. *Annals of Mathematics and Artificial Intelligence* (to appear)

[Fitting 1989] M.C. Fitting (1989) Bilattices and the theory of truth. *J. of Philos. Logic*, **18**: 225–256.

[Froidevaux and Mengin 1994] C. Froidevaux and J. Mengin (1994) Default logics: a unified view. *Computational Intelligence* **10**: 331–369.

[Gelfond et al. 1991] M.Gelfond, V.Lifschitz, H. Przymusińska and M. Truszczyński (1991) Disjunctive defaults. In *Proc. Second Int. Conf. on Principles of Knowledge Representation and Reasoning, KR'91*, Cambridge, MA., pp. 230–237.

[Goldszmidt and Pearl 1991] M.Goldszmidt and J. Pearl (1991) System Z^+: A formalism for reasoning with variable strength defaults. *Proc. AAAI-91*.

[Inoue and Sakama 1994] K. Inoue and C. Sakama (1994) On positive occurrences of negation as failure. In *Proc. 4th Int. Conf. on Principles of Knowledge Representation and Reasoning, KR'94*, Morgan Kaufmann, San Francisco, CA., 1994, pp. 293–304.

[Kraus et al. 1990] S. Kraus, D. Lehmann and M. Magidor (1990) Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, **44**: 167–207.

[Levesque 1984] H.J. Levesque. A logic of implicit and explicit belief. In *Proc. AAAI-84*, Austin, TX.

[Lifschitz 1994] V. Lifschitz (1994) Minimal belief and negation as failure. *Artificial Intelligence*, 70:53–72, 1994.

[Lifschitz and Woo 1992] V. Lifschitz and T. Woo (1992) Answer sets in general nonmonotonic reasoning (preliminary report). In *Proc. Third Int. Conf. on Principles of Knowledge Representation and Reasoning, KR'92*, Morgan Kaufmann, 1992, pages 603–614.

[Lin and Shoham 1992] F. Lin and Y. Shoham (1992) A logic of knowledge and justified assumptions. *Artificial Intelligence*, 57:271–289, 1992.

[Lukaszewicz 1988] W. Lukaszewicz (1988) Considerations on default logic: an alternative approach. *Computational Intelligence* **4**: 1–16.

[Marek et al. 1993] V. W. Marek, G. F. Schwarz, and M. Truszczyński Modal nonmonotonic logics: ranges, characterization, computation. *J. ACM*, 40:963–990, 1993.

[Moore 1985] R. C. Moore (1985) Semantical considerations on non-monotonic logic. *Artificial Intelligence*, **25**:75–94.

[Pearl 1990] J. Pearl (1990) System Z: A natural ordering of defaults with tractable applications to nonmonotonic reasoning. In M. Vardi (ed.) *Proc. TARK'90*, Morgan Kaufmann, pp. 121–135.

[Pereira and Alferes 1992] L.M. Pereira and J.J. Alferes (1992) Well-founded semantics for logic programs with explicit negation. In B. Neumann (ed.) *Proc. European Conf. on AI*, John Wiley & Sons, pp. 102–106.

[Przymusinska and Przymusinski 1994] H. Przymusinska and T. Przymusinski (1994) Stationary Default Extensions. *Fundamenta Informaticae*, **20** (in print).

[Przymusinski 1991] T.C. Przymusinski (1991) Stable semantics for disjunctive programs *New Generation Computing*, **9**: 401–424.

[Przymusinski 1995] T.C. Przymusinski (1995) Static semantics for normal and disjunctive logic programs. *Annals of Mathematics and Artificial Intelligence*

[Sakama and Inoue 1995] C. Sakama and K. Inoue (1995) Paraconsistent stable semantics for extended disjunctive programs. *J. of Automated Reasoning*, **5**: 265–285.

[Schaub 1991] T. Schaub (1991) On commitment and cumulativity in default logic. In *Symbolic and quantitative approaches to uncertainty*. LNCS 548, Springer, pp. 305–309.

[Schwarz 1992] G. Schwarz (1992) Minimal model semantics for nonmonotonic modal logics. *Proc. LICS-92*, Santa Cruz, CA., pp. 34–43.

[Schwarz and Truszczyński 1994] G. Schwarz and M. Truszczyński (1994) Minimal knowledge problem: a new approach. *Artificial Intelligence* **67**: 113–141.

Is there a logic of provability for nonmonotonic reasoning?

Gianni Amati
Fondazione Ugo Bordoni
via Baldassarre Castiglione, 59
00142 Roma, Italia
e-mail: gba@fub.it

Fiora Pirri
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
via Salaria, 113
00198 Roma, Italia
e-mail:pirri@assi.dis.uniroma1.it

Abstract

Nonmonotonic logic is a well developed research area gathering the logical formalisms that handle commonsense reasoning. One of the central issue in the area is to establish a notion of nonmonotonic provability, which is usually characterized via some modal logic. In [MST93] a clear paradigm has been proposed that classifies modal monotonic logics, augmented with the "negation by failure" to prove rule. It turns out that it is not the only one. In this paper we propose different paradigms for formalizing the notion of nonmonotonic provability. We present a fixed point construction allowing the interpretation of the necessity modal operator as a nonmonotonic provability operator analogously to that used in the modal logic of provability for Peano Arithmetic. We show that it faithfully interprets both default and autoepistemic logics and we highlight its connections with the standard fixed point construction *à la* McDermott and Doyle. This new paradigm provides a different perspective in the classification of modal nonmonotonic logics.

1 Introduction

Nonmonotonic logic is a central issue in Knowledge Representation. It investigates the reasoning of an agent interacting with the real world in which she often has to take the troublesome decision of completing or withdrawing information in order to be coherent with her observations and undertake suitable actions.

Following the first formalisms introduced by Reiter [Rei80], McDermott and Doyle [MD80], McCarthy [McC80, Lif85] and Moore [Moo87, Moo85] many researchers proposed new nonmonotonic logics, as variants of the earlier (see e.g [Bre91, Dix92, FM94, GM94, Luk88] for alternatives to Reiter's default logic). After many efforts in this direction it turned out that the main problem was not to invent new formalisms. The real point was, indeed, that of understanding and making clear the concept of *nonmonotonic provability* yet implicit in the earliest formalisations. In fact, nonmonotonic logic is a context dependent form of reasoning requiring context dependent proofs. The natural way to express the meta logical concepts involved is to turn to modal logics, where notions like "it is provable" or "it is consistent to assume" can be dealt with via the modal operator.

The main contribution in this direction comes from the work of Marek, Schwartz and Truszynszcki (MST) who provided a thorough treatment of non monotonic logics. They understood that McDermott and Doyle (MC&D) construction could represent influential nonmonotonic logics – autoepistemic logic, defaults and stable set semantics for logic programs [GL88] – and envisage new nonmonotonic logics. The very simple idea is that MD&D paradigm embeds "negation by failure to prove" into monotonic modal logics in so establishing the concept of *stable expansion*, which is the determining principle of nonmonotonicity:

$$\mathcal{T} = Cn_{\Lambda}(I \cup \{\neg\Box\varphi : \varphi \notin \mathcal{T}\}) \qquad (1)$$

where Λ is a (modal) logic, I is a set of assumption in the modal language and $\mathcal{T}$ is said to be a stable expansion of I if Equation (1) is satisfied. The negation by failure paradigm is expressed in braces: the non provability of φ is added whenever φ cannot be proved.

The very important contribution of MST is that of introducing *ranges* of logics [MST93], where a range

is a collection of monotonic modal logics generating the same concept of consistent expansion. The significance of the notion of *range* relies on the fact that it provides a simplification on the universe of nonmonotonic logics as it presents monotonic classes of logics under which nonmonotonic formalisms can be understood. In fact, the notion of range shows that there are classes of modal logics which behave similarly in the presence of negation by failure, thus collapsing into the same nonmonotonic logic. Recently Schwartz has introduced a notion, for a logic, to be *maximal* in a range [Sch95].

The deal is, now, that of completing the picture of these classes and of the schemata of nonmonotonic equations, like (1) above. The usefulness of this sort of taxonomisation, as we said, relies on a simplification of the nonmonotonic universe (and more generally of the logics used to represent knowledge and beliefs) and also it helps to provide a definition of the concept of nonmonotonic provability and to clarify which is the most pliable modal logic. This classification of monotonic modal logics with respect to some nonmonotonic schema that is based on the sole "maximality in a range" notion, however does not take into account an important criterion: to what extent a maximal nonmonotonic modal logic is expressive enough to embed different nonmonotonic formalisms. A maximal nonmonotonic modal logic which is able to "naturally" capture different formalisms should be accredited as the good one for treating any combination of different forms of commonsense reasoning.

Under the view just presented, in this work we are concerned with two major issues: the first is related to the classification given by MTS. Furthermore, we identify a different fixed point equation which gives different ranges of logics. For example, both default and autoepistemic logic have a simple translation in any nonmonotonic modal logic of a certain class. More precisely, we shall discuss the following points:

1.a $KD4Z$ is in the same range of $KD45$ with respect equation (1).

1.b By slightly modifying equation (1), that is, introducing the notion of *boxed* fixed point (see Section 3 below) and giving a somehow different embedding of default logic in modal logic, we show that a new class of modal logics Λ, $KD4 \leq \Lambda \leq KD45.1$ capture default logic, while any modal logic in Λ, $KD4Z \leq \Lambda \leq KD45.1$ can embed both default and autoepistemic logic.

On the other hand, the second issue deals with the following question "to which extent negation by failure can be simulated in the modal language itself?" We explore this point following the idea realised by [Ber75, Sam75, Sol76] for representing provability in Peano arithmetic with the modal logic G. Already McDermott and Doyle speculated in their paper (see the Discussion in [MD80]) that the modal logic G could be applicable in improving their nonmonotonic logic, but they never carried through in proving this connection. As a matter of fact G is not suitable for "nonmonotonic provability" while $KD4Z$ is (for a comparison between G and $KD4Z$ on this argument see [ACP96a]). We treat the necessity operator $\Box$ as a *nonmonotonic provability* operator by adding to each boxed formula a context, as a new propositional variable. A context, thus, becomes part of the language and it is used as a parametric theory. We resume the problems concerned with contextual and self-referential reasoning in the following points:

2.a We discuss how contexts can be represented in the language thus yelding self-referential sentences in which both the notion of *it is provable from the context* and *it is consistent with context* can be suitably represented in the modal logic.

2.b We show that default extensions can be represented by means of self-referential statements (a one to one correspondence between internal and external fixed points can be established in the language).

2.c Also autoepistemic logic has a "representation theorem" through internal fixed points, though the above mentioned correspondence is not one to one. However we show that with the internalisation of the context into the language, there is a paradigm – extending MD&D – that allows any logics – extending $K4$ – to embed autoepistemic logic still preserving its deductive power.

Some interesting considerations follow from the results we are going to present. There are different paradigms for nonmonotonicity and, more interesting, there are different logical schema which arise from the addition of negation by failure to modal monotonic logics. A crucial one is that dealing with the contextualisation of the "provability" operator. We show that ranges and, thus, classes of modal monotonic logics yelding nonmonotonic logics, are sensible to the paradigm chosen. Although the picture is far from being completed, this paper adds some nice tassels to the puzzle of "what monotonic modal logic is able to perform best and most of nonmonotonic forms of nonmonotonic reasoning".

2 Preliminaries

In the sequel, $\mathcal{L}$ and $\mathcal{L}_\Box$ denote, respectively, the sets of well formed formulae of propositional logic PC and of the modal language obtained by extending $\mathcal{L}$ with the $\Box$ operator, where $\Diamond = \neg\Box\neg$. With Λ we denote any modal logic. We recall that the minimal ***subnormal modal logic*** N contains the tautologies of PC, and is closed under the universal substitution rule US, modus ponens MP and necessitation Nec; K is the minimal ***normal modal logic*** which extends N with the normality axiom $\Box(\alpha \to \beta) \to (\Box\alpha \to \Box\beta)$. Axiom schemata of Λ can be:

D:	$\Box\varphi \to \Diamond\varphi$	seriality
4:	$\Box\varphi \to \Box\Box\varphi$	transitivity
T:	$\Box\varphi \to \varphi$	reflexivity
5:	$\Diamond\Box\varphi \to \Box\varphi$	euclideaness
5.n:	$\Diamond\Box\varphi \to \Box\Box^n\varphi$	n– euclideaness with $n \geq 1$
Z:	$\Box(\Box\varphi \to \varphi) \to (\Diamond\Box\varphi \to \Box\varphi)$	discreteness
f:	$(\Diamond\Box\varphi \wedge \psi) \to \Box(\varphi \vee \Diamond\psi)$	
$w5$:	$\Diamond\Box\varphi \to (\varphi \vee \Box\varphi)$	
W:	$\Box(\Box\varphi \to \varphi) \to \Box\varphi$	finiteness

If $S_1, \ldots, S_n$ are schemata then $(K)S_1 \ldots S_n$ is the (normal) modal logic generated by $(K)S_1, \ldots, S_n$. A sentence φ is ***provable*** in the logic $\Lambda = (K)S_1 \ldots S_m$, denoted $\vdash_\Lambda \varphi$ if it has a proof from $(K), S_1, \ldots, S_m$. That is, there is a sequence of formulae $\varphi_0, \ldots, \varphi_n = \varphi$ and, for each i, $i \leq n$, either φ_i is a propositional tautology, or is a substitution instance of $(K), S_1, \ldots, S_m$, or it has been obtained by MP or by Nec. The set of theorems of Λ coincides with the set of formulae in Λ that have a proof from $(K), S_1, \ldots, S_m$. By $I \vdash_\Lambda \varphi$ we denote a formula φ having a Λ-proof from I. $Cn_\Lambda(I) = \{\alpha | I \vdash_\Lambda \alpha\}$, with $I \subseteq \mathcal{L}_\Box$, denotes the set of formulae containing I and deductively closed under the modal logic Λ. For two logics Λ_1 and Λ_2 by $\Lambda_1 \leq \Lambda_2$ we mean that the set of theorems of Λ_1 is contained in that of Λ_2. A set of formulae S is said to be ***boxed*** if any formula φ in S is boxed, i.e. any propositional variable occurring in φ is in the scope of a modal operator. We refer the reader to [Gol87] for the basics of modal logic.

For any normal modal logic Λ the following deduction theorem holds (see [Fit83]): $\varphi \in Cn_\Lambda(I)$ iff there is a finite subset J of I, and an integer n, such that $\vdash_\Lambda A \wedge \Box A \wedge \ldots \wedge \Box^n A \to \varphi$ where A is the logical conjunction of the elements of J. If $\Lambda \geq K4$, then $\varphi \in Cn_\Lambda(I)$ iff there is a finite subset J of I, such that $\vdash_\Lambda A \wedge \Box A \to \varphi$ where A is the logical conjunction of the elements of J.

We shall assume some familiarity with the results of [Rei80, Moo85, Kon88, MST93]. We briefly recall, however, that a stable theory is a set of formulae $\mathcal{T}$ closed under PC and Nec and the rule "if $\alpha \notin \mathcal{T}$ then $\neg\Box\alpha \in \mathcal{T}$". If S is a set of propositional formulae, by $St(S)$ we denote the unique stable theory $\mathcal{T}$ such that $\mathcal{T} \cap \mathcal{L} = Cn_{PC}(S)$ [MT93]. Given a theory $I \subseteq \mathcal{L}_\Box$ a theory $\mathcal{T} \subseteq \mathcal{L}_\Box$ is a Λ expansion of I iff $\mathcal{T}$ satisfies the equation:

$$\mathcal{T} = Cn_\Lambda(I \cup \neg\Box\overline{\mathcal{T}}) \tag{2}$$

where $\neg\Box\overline{\mathcal{T}} = \{\neg\Box\alpha : \mathcal{T} \not\vdash_\Lambda \alpha\}$.

The necessity operator $\Box$, is in general given the notion of ***knowledge*** or ***belief***, here we shall mainly interpret it as "it is provable". E.g. $\Box\alpha$ means that α is a theorem, on the other hand $\neg\Box\alpha$ means that α is not a theorem, where these concepts are used in relation to a given theory, or context.

In addition to the logics used in [MST93] we shall also consider the logics $KD4Z$ and $KD45.n, n \geq 1$. $KD4Z$ is the modal logic obtained extending $KD4$ with the discreteness axiom schema Z. Axiom schema Z says that a formula α is a theorem if both it is provable its soundness with respect to truth, i.e. $\Box(\Box\alpha \to \alpha)$, and is not provable its unprovability, i.e. $\Diamond\Box\alpha$. Indeed, axiom schema Z "embedds" axiom schema 5, as soon as $\Box(\Box\alpha \to \alpha)$ is true, i.e. as soon as reflexivity, i.e $\Box\alpha \to \alpha$, becomes provable; in so explaining the frame structure of this logic which results from a concatenation $\mathcal{F} \bigodot \mathcal{C}$ of an irreflexive frame $\mathcal{F}$ of finite-depth and of a universal frame, i.e. a cluster $\mathcal{C}$ (see [MT93, ACGP96] for details on this notation involving Kripke frames and [ACGP96] for a semantic characterization of $KD4Z$). A derived rule of $KD4Z$ is that α is a $KD4Z$-theorem iff $\Box\alpha$ is a $KD4Z$-theorem. Note that this equivalence implies the decidability of α as far as $\Box$ acts as a provability operator. For a discussion on the role of $KD4Z$ as a provability logic and its relation with the 'classical' provability logic $G = KW$ ([Boo79, Sol76]), see [ACP96a].

The other new modal logic that we deal with is $KD45.n$. It extends $KD4$ with the n-euclideaness axiom. It says that if a formula α can ever be proved then it will certainly before the "step" $n+1$ but, differently from Z, it says nothing about the truth of α. Obviously $KD45.n$ contains $KD4Z$. With a proof similar to the one for $KD4Z$ (see [ACGP96]) it is easy to show that theorems of $KD45.n$ are formulae valid in the set of frames which result from the concatenation $\mathcal{F} \bigodot \mathcal{C}$ of two frames, where $\mathcal{F}$ is of depth $n+1$ and $\mathcal{C}$ is a cluster.

A terminology related to contextual sentences follows. By $E(p)$ we denote a formula in which the proposi-

tional variable p occurs and by $E(\varphi)$ the formula in which the formula φ has been uniformly replaced for p. Sometimes we shall write $E(C)$ to stress that the variable C denotes a context in the formula. For example if $E(C)$ is $\Box(C \rightarrow \gamma)$ and C denotes the theory $\Box(\alpha \wedge \beta)$ then the substitution of $\Box(\alpha \wedge \beta)$ for C in $E(C) = \Box(C \rightarrow \gamma)$ will give $\Box(\Box(\alpha \wedge \beta) \rightarrow \gamma)$. C is, thus, used as a parametric theory, i.e. it denotes the theory which, substituted for it in $E(C)$, will return a formula in which the theory itself is realised.

3 Ranges, expansions and nonmonotonic rules

The modal approach of MD&D fixed point has been taken as a paradigm to introduce a proof-theoretic expressiveness in the language, according to the notion of contextual proof. The idea behind the modal approach is to adopt negative introspective reasoning via the negation by failure to prove rule (NBF), which consists in the explicit assumption of all those sentences stating the unprovability of those formulae not belonging to the context. NBF can be defined as the following context dependent rule:

$$\frac{A \notin \mathcal{T}}{\neg\Box A} \tag{3}$$

In other words, the above rule says that if A does not belong to the context, i.e. it is not provable with the available information, then state that A is not a theorem. In the presence of NBF, the modal contexts, which are expansions, turn to be stable sets. So, we may suppose $\mathcal{T}$ stable when using NBF. Let Λ be a modal logic, $\mathcal{T}$ a consistent modal theory and I an initial set of assumptions. The *nonmonotonic deductive closure* $Cn^{\mathcal{T}}_{\Lambda}(I)$ *of* I *with respect to a context* $\mathcal{T}$ is the deductive closure of $I \cup \{\neg\Box\overline{\mathcal{T}}\}$. In other words, the logic Λ is extended to $\Lambda^{\mathcal{T}}$ by adding NBF. MTS classification aims at identifying families of monotonic modal logics which, extended with NBF, yeld the same nonmonotonic logic. A nonmonotonic logic $\Lambda^{\mathcal{T}}$ belongs to a range $\Lambda_1 \leq \Lambda \leq \Lambda_2$ if every monotonic modal logic in the family yelds the same Λ-expansion. MTS identify many of such ranges and recently Schwarz has shown that, indeed, infinitely many non equivalent nonmonotonic modal logics can be obtained by varying the choice of the underlying monotonic modal logic. Let us say that $\Lambda^{\mathcal{T}}$ is a *basic nonmonotonic logic* established on Λ with context $\mathcal{T}$. What does it happen if we enrich the notion of Λ-expansion, that is, if we add to the monotonic logic Λ context dependent rules other than NBF? It turns out that in some case new rules bring to the same concept of Λ-expansions based on NBF. Let us present the following rules:

$$\frac{A \in \mathcal{T}}{\Box A} \quad (\Box^{\mathcal{T}})$$

$$\frac{A \in \mathcal{T}}{\Box A \leftrightarrow A} \quad (Eq^{\mathcal{T}})$$

$$\frac{A \in \mathcal{T} \quad \Box A \rightarrow A}{A} \quad (L\ddot{o}b^{\mathcal{T}})$$

$$\frac{A \in \mathcal{T} \quad \Box(\Box A \rightarrow A)}{\Box A} \quad (Z^{\mathcal{T}})$$

In general $\mathcal{T}$ is said to be a Λ-*expansion* if $Cn^{\mathcal{T}}_{\Lambda}(I) = \mathcal{T}$. Analogously, we say that $\mathcal{T}$ is a Λ-X-*expansion* if $Cn^{\mathcal{T}}_{\Lambda}(I) = \mathcal{T}$ and $\Lambda^{\mathcal{T}}$ is enriched by the rule X.

Moore [Moo85] has proved that if Λ is classical propositional logic then Λ-$\Box^{\mathcal{T}}$-expansions are equivalent to $KD45$-expansions. Observe also that classical expansions with $Eq^{\mathcal{T}}$ rule are equivalent to $Sw5$-expansions and $KD45$-expansions can be embedded into $w5$-expansions, and vice versa (see [MST93]).

Each of the above rules are modal rules depending on a context. In particular, the $L\ddot{o}b^{\mathcal{T}}$ rule can be understood as a way of extending the applicability of the Löb rule [Smo85]

$$\frac{\Box A \rightarrow A}{A} \tag{4}$$

from the theorems A, of the logic, to the formulae of $\mathcal{T}$. Indeed in G the Löb rule says that a formula A is a theorem iff $\Box A \rightarrow A$ is. The Löb rule is used in the modal logic $KW = G$ and is equivalent to the axiom $W = \Box(\Box A \rightarrow A) \rightarrow \Box A$.

Analogously, the $Z^{\mathcal{T}}$ rule can be understood as an extension of the following rule

$$\frac{\Diamond\Box A \quad \Box(\Box A \rightarrow A)}{\Box A} \tag{5}$$

which is implied by axiom schema Z. The $Z^{\mathcal{T}}$ rule, is obtained from (5) by the fact that $A \in \mathcal{T}$ is equivalent to $\Diamond\Box A$ and $A \in \mathcal{T}$, via NBF, if $\mathcal{T}$ is stable and consistent.

It is easy to show that the $L\ddot{o}b^{\mathcal{T}}$ rule is equivalent to the $Z^{\mathcal{T}}$ rule, whenever $\mathcal{T}$ is stable and consistent. Furthermore, $\Box^{\mathcal{T}}$ rule implies the $L\ddot{o}b^{\mathcal{T}}$ rule; in fact, suppose $A \in \mathcal{T}$ and $\Box A \rightarrow A$, then from the $\Box^{\mathcal{T}}$ we get $\Box A$ and, by MP, we get A.

On the other hand, in a basic nonmonotonic logic $\Lambda^{\mathcal{T}}$, with $\Lambda \geq K4$, axiom schema Z is equivalent to $Z^{\mathcal{T}}$

rule whenever $\mathcal{T}$ is stable and consistent. In fact, assume axiom schema Z, $A \in \mathcal{T}$ and $\Box(\Box A \rightarrow A)$, since $\mathcal{T}$ is, by hypothesis, stable and consistent then $\Diamond\Box A$ follows by NBF and, by Z and $\Box(\Box A \rightarrow A)$, $\Box A$ follows by MP. For the other direction, observe that $\vdash_{K4} \Box B \rightarrow B$ with $B = \Box(\Box A \rightarrow A) \rightarrow \Box A$ (see [Smo85] page 76). If $A \notin \mathcal{T}$ then $\neg\Diamond\Box A$, hence Z follows from a classical tautology. If $A \in \mathcal{T}$ then $\Diamond\Box A$ is provable and $\Box A \in \mathcal{T}$ which implies $B \in \mathcal{T}$. From the $L\ddot{o}b^{\mathcal{T}}$ rule we get B, then $\Diamond\Box A$ and B imply Z.

This proves

Theorem 1 Let $K4 \leq \Lambda$. $\mathcal{T}$ is a Λ-$Z^{\mathcal{T}}$-expansion iff $\mathcal{T}$ is a ΛZ-expansion.

On the other hand $\Box^{\mathcal{T}}$ rule is equivalent to 5 whenever $\mathcal{T}$ is stable and consistent. In fact, assume $\Box^{\mathcal{T}}$ and $\alpha \in \mathcal{T}$ hence $\Box\alpha$ follows. By the tautology $\Box\alpha \rightarrow (\Diamond\Box\alpha \rightarrow \Box\alpha)$ and MP, axiom schema 5 follows. If $\alpha \notin \mathcal{T}$, then, by NBF, we get $\neg\Box\alpha$ and, by Nec, we get $\Box\Diamond\neg\alpha$. From tautology $\Box\Diamond\neg\alpha \rightarrow (\Diamond\neg\alpha \rightarrow \Box\Diamond\neg\alpha)$, 5 follows by MP. The other direction is obvious.

Therefore, if $\Lambda 5$ is any modal logic extending the basic non-normal modal logic N with axiom schema 5, then:

$$Cn^{\mathcal{T}}_{\Lambda 5}(I) \equiv \mathcal{T} \text{ if and only if } Cn^{\mathcal{T}}_{\Lambda}(I \cup \Box\mathcal{T}) \equiv \mathcal{T} \quad (6)$$

Let us now turn to the characterization given by MTS of default and autoepistemic logic, we have the well known results that all Λ's with $5 \leq \Lambda \leq KD45$ are in the same range of AEL-expansions and that translating a default theory $\langle W, D\rangle$ as

$$I = W \cup \bigwedge \Box\alpha \wedge \Box\Diamond\beta \rightarrow \gamma \quad (7)$$

for each $\alpha : \beta/\gamma$ in D, any Λ in $N \leq \Lambda \leq S4f$ yelds Reiter extensions.

It turns out that as far as autoepistemic logic is concerned, the above range is not complete as the following result shows:

Theorem 2 $\mathcal{T}$ is a $KD45$-expansion if and only if $\mathcal{T}$ is a $KD4Z$-expansion.

Proof. We prove the hard direction. Marek and Truszczynski proved that (see [MT93] Corollary 9.8) $\mathcal{T}$ is an expansion if and only if $\mathcal{T}$ is stable and consistent and $\mathcal{T} \subseteq Cn^{\mathcal{T}}_{\Lambda}(I)$. That is for $\mathcal{T}$ consistent and stable, $I \subseteq \mathcal{T}$ if $\mathcal{T} \subseteq Cn^{\mathcal{T}}_{KD45}(I)$ then $\mathcal{T} \subseteq Cn^{\mathcal{T}}_{KD4Z}(I)$. Since $KD45$ extends $KD4Z$ then $\mathcal{T} = Cn^{\mathcal{T}}_{KD45}(I)$ iff $\mathcal{T} = Cn^{\mathcal{T}}_{KD4Z}(I)$.

From (6) and by the deduction theorem we have that for any $A \in \mathcal{T}$ there is a finite set of formulae $A^i \in \mathcal{T}$, $1 \leq i \leq n$, such that $I \cup \neg\Box\overline{\mathcal{T}} \vdash_{KD4} \bigwedge_{1\leq i\leq n} (\Box\Box A^i \wedge \Box A^i) \rightarrow A$, hence by transitivity $I \cup \neg\Box\overline{\mathcal{T}} \vdash_{KD4} \bigwedge_{1\leq i\leq n} \Box A^i \rightarrow A$. Let A_1 be $\bigwedge_{1\leq i\leq n} A^i$. By axiom K, $I \cup \neg\Box\overline{\mathcal{T}} \vdash_{KD4} \Box A_1 \rightarrow A$. Let us iterate the reasoning with A_1 instead of A. A_1 is in $\mathcal{T}$, and thus we obtain an infinite chain of implications $I \cup \neg\Box\overline{\mathcal{T}} \vdash_{KD4} \Box A_{n+1} \rightarrow A_n$, $n \geq 1$ and $A_n \in \mathcal{T}$.

Let us extend $KD4$ with the axiom Z. Suppose, per absurdum, that $I \cup \neg\Box\overline{\mathcal{T}} \nvdash_{KD4Z} \Box A \rightarrow A$, then there is a $KD4Z$-model of $I \cup \neg\Box\overline{\mathcal{T}}$ such that $w \models \Box A \wedge \neg A$. Since $w \models \Box A_1 \rightarrow A$, then we may find a world w_1, wRw_1 with $w_1 \models \neg A_1$. We can thus build an infinite ascending chain of worlds $wRw_1Rw_2 \ldots Rw_nR \ldots$ such that $w_n \models \neg\Box A_{n+1} \wedge \neg A_n$. Thus there is an integer n such that all w_ms with $m \geq n$ are in the last cluster. Since every A_i is in $\mathcal{T}$ and for all i we have $\Diamond\Box A_i \in \neg\Diamond\overline{\mathcal{T}}$, we thus get the inconsistency because this would imply that for all w_ms, with $m \geq n$, $w_m \models \Box A_i$ should hold. Hence $I \cup \neg\Box\overline{\mathcal{T}} \vdash_{KD4Z} \Box A \rightarrow A$, and by necessitation and axiom Z we get $I \cup \neg\Box\overline{\mathcal{T}} \vdash_{KD4Z} \Diamond\Box A \rightarrow \Box A$. Again by the fact that $\mathcal{T}$ is stable and consistent we have $\Diamond\Box A$ then, by modus ponens, $\Box A$ and finally A. □

This implies that axiom schema 5 is not the real "culprit" to distinguish autoepistemic logic. Therefore we can enlarge the range for autoepistemic logic as follows:

Corollary 1 Let Λ be a modal logic. If $KD4Z \leq \Lambda \leq KD45$ the McDermott fixed point construction yelds autoepistemic logic.

4 New paradigms

So far we dealt with the fixed point construction given in equation (1). It is natural to ask whether it is the only possible tool for defining nonmonotonic logics. We show that several alternatives can be chosen. We shall, indeed, give three new paradigms:

i. Boxed and super boxed expansions;

ii. The self-reference paradigm;

iii. The contextual expansion paradigm.

We shall present these new paradigms in the next three sections.

4.1 Boxed and super boxed expansions

A *boxed fixed point* equation for a set I of boxed formulae is of the form

$$Cn_\Lambda(I \cup \neg\Box\overline{\mathcal{T}}) = Cn_\Lambda(\Box\mathcal{T}) \tag{8}$$

Any modal theory $\mathcal{T}$ which is a solution to this equation is called a *boxed expansion*.

The basic idea of boxed expansions is to treat all formulae of a context as theorems, namely as sentences asserting the provability of other sentences. It turns out that the set of assumptions I in (8) can be treated as set of beliefs; in the sense that truth and provability can be clearly distinguished and further combined. For example, $\Box A \rightarrow A$ is the belief "if $\Box A$ is true – i.e. $A \in \mathcal{T}$ – then A must be true". On the other hand $\Box A \rightarrow \Box B$ can be interpreted as "if $A \in \mathcal{T}$ then $B \in \mathcal{T}$". Obviously, the reflexivity axiom has not to hold in order to preserve independence between provability and truth. This subtle distinction has important consequences. Our conjecture, in fact, is that autoepistemic logic cannot be characterized by any of these kinds of nonmonotonic modal logics. A clear evidence is that default logic can be easily embedded in many of these nonmonotonic modal logics such as $KD45.1$, which is very close to the logic $KD45$ and Gottlob proved that no modular translation of default logic exists in AEL[Got93]). Nevertheless, it turns out that boxed fixed points are suitable to embed both autoepistemic and default logic under the same nonmonotonic modal logic Λ where $KD4Z \leq \Lambda \leq KD45.1$, as the following theorem states:

Theorem 3 Let $\mathcal{T}$ be consistent. $\mathcal{T}$ is an autoepistemic expansion of I iff $\mathcal{T}$ is a boxed expansion of $\Box I$ in $KD4Z$.

Proof. Let $\mathcal{T}$ be an autoepistemic expansion of I. By Corollary 1 we have that $\mathcal{T}$ is an expansion of I in $KD4Z$. Consider any model $\mathfrak{A}$ of $KD4Z$. Note that $\mathfrak{A} \models \mathcal{T}$ iff $\mathfrak{A} \models \Box I \cup \neg\Box\overline{\mathcal{T}}$, then $\mathfrak{A} \models \Box\mathcal{T}$ iff $\mathfrak{A} \models \Box I \cup \neg\Box\overline{\mathcal{T}}$. If we add a new initial world to $\mathfrak{A}$ and obtain another $KD4Z$ model $\mathfrak{A}'$, then we prove the other direction. □

For $KD45.n$ we can give a proof analogous to that of Theorem 3 by exploiting the irreflexivity of $\mathcal{F}$ in the $KD45.n$ frame $\mathcal{F} \odot \mathcal{C}$, and Corollary 1.

Theorem 3 says that $\mathcal{T}$ is an autoepistemic expansion of I whenever $\mathcal{T}$ is consistent and satisfies the following equation:

$$Cn_\Lambda(\Box I \cup \neg\Box\overline{\mathcal{T}}) = Cn_\Lambda(\Box\mathcal{T}) \tag{9}$$

We call (9) a *super boxed fixed point.*

Turning to Reiter's default logic, the notion of context is used to derive nonmonotonic consequences according to the way justifications are managed by nonmonotonic inference rules. Besides the results of MTS presented in the previous section, Truszczynski [Tru91] considers also the following boxed translation:

$$tr(\langle W, D\rangle) = \Box W \cup \{\Box\alpha \wedge \Box\Diamond\beta \rightarrow \Box\gamma |\ \alpha : \beta/\gamma \in D\} \tag{10}$$

where $\langle W, D\rangle$ is a default propositional theory. Truszczynski shows that any subnormal logic Λ with reflexivity and contained in $S4f$, that is $T \leq \Lambda \leq S4f$ yelds Reiter extensions, under the above translation. We translate defaults in modal logic according to the intuition that all formulae involved in our reasoning must be "provable" or "consistent". Our translation is, in fact, the same as (10).

If $\Lambda \leq S5$, a boxed expansion $\mathcal{T}$ is then a stable set containing I. Let I denote $tr(\langle W, D\rangle)$, we give the following completeness theorem for Reiter's default logic.

Theorem 4 (COMPLETENESS) Let $\langle W, D\rangle$ be a default theory. Let $KD4 \leq \Lambda \leq KD45.1$ and $\mathcal{T}$ be a consistent stable set.

$$\text{If } Cn_\Lambda(I \cup \neg\Box\overline{\mathcal{T}}) = Cn_\Lambda(\Box\mathcal{T})$$
$$\text{then } \mathcal{L} \cap \mathcal{T} \text{ is a Reiter extension}$$

Proof. Let $\mathcal{T}$ be a boxed expansion for $I = \Box W \cup \{\Box\alpha \wedge \Box\Diamond\beta \rightarrow \Box\gamma |\ \alpha : \beta/\gamma \in D\}$. First we show that $\mathcal{L} \cap \mathcal{T}$, is closed with respect to defaults of D. Let $\alpha : \beta/\gamma$ be in D, $\alpha \in \mathcal{T}$ and $\neg\beta \notin \mathcal{T}$. If $\mathfrak{C}$ is the universal $S5$- model corresponding to the stable set of $\mathcal{L} \cap \mathcal{T}$ we have to prove that $\mathfrak{C} \models \Box\gamma$. Now $\mathfrak{C} \models \Box\Diamond\beta$, while $\alpha \in \mathcal{T}$ implies $\Box\alpha \in \mathcal{T}$ because, by Corollary 4, $\mathcal{T}$ is stable, hence $\mathfrak{C} \models \alpha$. Since $\mathfrak{C}$ is a $KD4Z$ model and $\mathfrak{C} \models \Box\mathcal{T}$, then $\mathfrak{C} \models I$, that is $\mathfrak{C} \models \Box\gamma$ hence $\gamma \in \mathcal{T}$.

Let $\mathfrak{A} = \langle W, R\rangle$ be any $KD45.1$ model of $\mathcal{T}$ where the terminal cluster V is a set of worlds satisfying $\mathcal{T}$ and w_0 is the initial world. Consider the set $S = \bigcup_{n<\omega} S_n$ where S_0 is W and $S_n = \{\gamma \mid (\bigcup_{k<n} S_k) \vdash_\Lambda \alpha$ where $\alpha : \beta/\gamma \in D$, β consistent with $\mathcal{T}\}$. If we show that S is $\mathcal{L} \cap \mathcal{T}$ then we prove that $\mathcal{L} \cap \mathcal{T}$ is a Reiter extension (S is the smallest set of conclusions of the default proofs with $\mathcal{L} \cap \mathcal{T}$ used as context). Since $\mathcal{L} \cap \mathcal{T}$ is closed w.r.t. the set of defaults of D, and $W \subseteq \mathcal{L} \cap \mathcal{T}$ then, by induction on the integer n, $S \subseteq L \cap \mathcal{T}$.

Let U be the set of worlds satisfying S. Let us consider the model $\mathfrak{A}' = \mathfrak{A} \cup \{w_0 R u, URV \mid u \in U$ and V is the

terminal cluster} and extends the satisfiability relation on the formulae over U, by forcing. If $\mathfrak{A}$ is a universal model make a copy of an element of U and put it as first element. $\mathfrak{A}'$ is a $KD45.1$ model. We have $\mathfrak{A}' \models \neg\Box\overline{\mathcal{T}}$. If $\mathfrak{A}' \models \Box\mathcal{T}$ then $U \models \mathcal{T}$ hence $S \vdash_\Lambda \mathcal{T} \cap \mathcal{L}$, which is the claim. Then suppose $\mathfrak{A}' \not\models \Box\mathcal{T}$, hence $\mathfrak{A}' \not\models I$. For all defaults $\alpha : \beta/\gamma \in D$ making $\mathfrak{A} \not\models I$ there is a world w, $w \models \Box\alpha \wedge \Box\Diamond\beta$ and $w \models \neg\Box\gamma$. We must have wRU, by definition of $\mathfrak{A}'$, hence by irreflexivity of the worlds not in the terminal cluster V, for some u in U, $u \models \neg\gamma$ and $U \models \alpha$. Thus $S \vdash_\Lambda \alpha$. Hence there is a S_n such that $S_n \vdash_\Lambda \alpha$ and β is consistent with $\mathcal{T}$ then, by definition of S, $S_{n+1} \vdash_\Lambda \gamma$ which is a contradiction with $u \models \neg\gamma$. $\mathcal{L} \cap \mathcal{T}$ coincides with S, so $\mathcal{L} \cap \mathcal{T}$ is a Reiter extension. □

More generally:

Corollary 2 Let $KD4 \leq \Lambda \leq KD45.1$ or $KD4 \leq \Lambda \leq S4f$ and $\mathcal{T}$ a consistent stable set:

$$Cn_\Lambda(I \cup \neg\Box\overline{\mathcal{T}}) = Cn_\Lambda(\Box\mathcal{T})$$
$$\text{iff } \mathcal{L} \cap \mathcal{T} \text{ is a Reiter extension}$$

Observe, as interesting phenomena, that these new nonmonotonic modal logics seems to group the monotonic modal logics extending $KD4$ in "larger" families with respect the MD&D paradigm. At the same time the standard classification turns out not to be a refinement of this new one ($KD4Z$ and $KD45$ belong to different ranges in the boxed version while are in the same range with respect to the MD&D paradigm).

4.2 Self-reference for default and autoepistemic logic

In this section we discuss how to represent the relativisation of formulae of a given theory (either a default or an autoepistemic theory) to a context C, by interpreting the modal operator $\Box$ as a *nonmonotonic provability operator*. In other words a $\Box$ that binds a formula α is interpreted as "α is provable from the context C". Therefore, for instance, $\Box p$ is written as $\Box(C \rightarrow p)$.

If T is a default theory $\langle W, D\rangle$, we first get the modal theory $I = tr(\langle W, D\rangle)$ and then we contextualise it to $I(C)$. In other words, for each default $\alpha : \beta/\gamma$, embedded in $tr(\langle W, D\rangle)$ as $\Box\alpha \wedge \Box\Diamond\beta \rightarrow \Box\gamma$, we get the contextualised formula:

$$\Box(C \rightarrow \alpha) \wedge \Box(C \rightarrow \Diamond(C \wedge \beta)) \rightarrow \Box(C \rightarrow \gamma)$$

As a matter of fact, the nesting of contexts is not relevant with respect to the characterization we are going to give by means of fixed point theorems in the logic (see also next section for a discussion on this point). Observe that this is due not to the redundancy of nested beliefs, like in [Kon94] where the redundancy follows from the reduction of modalities in $KD45$; it is due, instead, to reductions that take place when substitutions of fixed points for contexts are performed. Therefore the above formula reduces to:

$$\Box(C \rightarrow \alpha) \wedge \Box\Diamond(C \wedge \beta) \rightarrow \Box(C \rightarrow \gamma)$$

Analogously, suppose we have the following autoepistemic clause: $\Box\alpha \rightarrow \alpha$, then, its contextualisation is

$$\Box(C \rightarrow \alpha) \rightarrow \alpha$$

In general, via the contextualisation, we get a formula $I(C)$ such that, if $\top$ is the constant representing a tautology,

$$I(\top) \equiv_\Lambda I$$

Indeed, we can treat a context in the language itself, because the logics we are dealing with provides a fixed point theorem. More precisely, a *fixed point* in a modal logic Λ for a modal predicate $E(C)$, in which the parameter C occurs boxed, is a modal formula $\mathcal{T}$ logically equivalent to $E(\mathcal{T})$:

$$\vdash_\Lambda \mathcal{T} \equiv E(\mathcal{T}) \tag{11}$$

The set of $\mathcal{T}$ satisfying (11) *explicitly* define the modal predicate $E(C)$. For example let

$$E(C) = \Box((\Box\Diamond(C \wedge A) \rightarrow \Box(C \rightarrow A)) \wedge \Diamond C) \tag{12}$$

The formula $\mathcal{T} = \Box(\Box\Diamond A \rightarrow \Box A)$ is in the set of formulae which explicitly define $E(\mathcal{T})$, since $\vdash_{KD4Z} E(\mathcal{T}) \equiv \mathcal{T}$. This interesting property is provided only by *implicit consistency predicates*, i.e. all those modal predicates $E(C)$ such that $E(C) \rightarrow \Box\Diamond C$. In other words, these modal predicates state the provability of their consistency. We have given in [ACP96a] a representation theorem for implicitly definable predicates, in $KD4Z$. The representation theorem allows us to obtain explicit definitions of implicit consistency predicates, in a mechanical way.

Let C be a propositional variable which occurs boxed in $I(C)$:

Theorem 5 Let $E(C) = \Box I(C)$ with $E(C) \rightarrow \Box\Diamond C$ and $\mathcal{T}$ a formula, then

$\vdash_{KD4Z} E(\mathcal{T}) \leftrightarrow \mathcal{T}$ iff $\vdash_{KD4Z} \mathcal{T} \leftrightarrow \Box I(\top) \wedge \Box \bigwedge_{i \in J} \Diamond \gamma_i$

with γ_i any formula and J a finite set of indexes.

The fixed points of $I(C)$ are partially ordered with respect to the relation $\alpha \leq \beta$ defined as $\vdash \alpha \rightarrow \beta$.

Example 1 if $E(C)$ is as in (12) then, besides $\mathcal{T}$, also $\Diamond\Box\neg A$, $\Box\Box A$ and $\bot$ are fixed points, and $\mathcal{T}$ is the top element, with respect to $\leq$, in the lattice of the foregoing mentioned fixed points.

On the basis of these results, the characterizations of default and autoepistemic logic follows below.

Theorem 6 Let I be a set of modal formulae and $E(C) = \Box I(C)$ its contextualised formula. Let

$$\vdash_{KD4Z} E(\mathcal{T}) \leftrightarrow \mathcal{T}$$

and $\mathcal{T}$ a minimal consistent fixed point. If $\mathcal{T} \subseteq St(A)$, where $A = \{\alpha | \mathcal{T} \vdash_{KD4Z} \Box\alpha, \alpha \text{ objective}\}$, then $St(A)$ is an autoepistemic expansion of I.

Proof. We show that $Cn_{KD4Z}(\Box St(A)) = Cn_{KD4Z}(\Box I \cup \{\Diamond\psi\}_{\{\neg\psi \notin St(A)\}})$. Then for Theorem 3 we have the claim. Let ψ be such that $\neg\psi \notin St(A)$. Then $\Box\Diamond\psi \in St(A)$. Since $\mathcal{T} \wedge \Box\Diamond\psi$ is also a fixed point then $\mathcal{T} \vdash_{KD4Z} \Box\Diamond\psi$ otherwise $\Box\Diamond\psi$ must be inconsistent with $\mathcal{T}$ for the minimality of $\mathcal{T}$, i.e. $\mathcal{T} \vdash_{KD4Z} \neg\Box\Diamond\psi$. From the hypothesis $\mathcal{T} \subseteq St(A)$ we get an inconsistency. We may suppose that $\mathcal{T} \vdash_{KD4Z} \Box\Diamond\psi$ for the minimality of $\mathcal{T}$. By induction on the complexity of the formulae in $St(A)$ and using the definition of A as first step of the induction, we get $Cn_{KD4Z}(\Box I \cup \{\Diamond\psi\}_{\{\neg\psi \notin St(A)\}}) = Cn_{KD4Z}(\Box St(A))$. □

Example 2 Let I be the autoepistemic theory $\Box\alpha \rightarrow \alpha$, $E(C)$ is $\Box(\Box(C \rightarrow \alpha) \rightarrow \alpha)$. A minimal fixed point is $\Box(\Box\alpha \rightarrow \alpha) \wedge \Diamond\Box\alpha$ ($\Diamond\Box\alpha$ is equivalent to $\Box\Diamond\Box\alpha$), then from the above characterization we get $\mathcal{T} = \Box\alpha$ and $St(\{\alpha\})$ is an autoepistemic expansion of I. On the other hand $\Box(\Box\alpha \rightarrow \alpha) \wedge \Box\Diamond\alpha \wedge \Box\Diamond\neg\alpha$ is also a minimal fixed point. In this case $St(\{\top\})$ is the autoepistemic expansion.

Let $\langle W, D\rangle$ be a default theory (where W is finitely axiomatisable and D is a finite set of defaults) and p a new propositional variable. $E(p)$ is associated with the default theory as follows:

$$E(p) = \Box(\Diamond p \wedge W \wedge I(p)) \tag{13}$$

where

$$I(p) = \bigwedge_{\alpha:\beta/\gamma \in D} \Box(p \rightarrow \alpha) \wedge \Box\Diamond(p \wedge \beta) \rightarrow \Box(p \rightarrow \gamma) \tag{14}$$

Observe that $\Diamond p$ in (13), is a requirement of consistency for a context. We define an additional condition Ψ which singles out the set of justifications of defaults in D maximally consistent with the parameter p as:

$$\Psi(p) = p \rightarrow \Box \bigwedge_{\beta \in J_D} (p \wedge \Diamond\beta \leftrightarrow p \wedge \Box\Diamond\beta) \tag{15}$$

The fixed point equation becomes:

$$p \leftrightarrow (E(p) \wedge \Psi(p)) \tag{16}$$

which is equivalent to

$$(p \leftrightarrow E(p)) \wedge \Psi(p) \tag{17}$$

by definition of $E(p)$.

We have the following characterization for default logic [ACP96a], where $Prov(\mathcal{T})$ denotes the set of the modal-free formulae α such that $\mathcal{T} \vdash_{KD4Z} \Box\Box\alpha$.

Theorem 7 Let $\vdash_{KD4Z} (\mathcal{T} \leftrightarrow E(\mathcal{T})) \wedge \Psi(\mathcal{T})$, $\mathcal{T}$ consistent, and $D' = \{\langle \frac{\alpha : \beta}{\gamma} \rangle : \beta \text{ consistent with } Prov(\mathcal{T})\}$. Then:

a) (REDUCT OF A FIXED POINT) $\vdash_{KD4Z} \mathcal{T} \leftrightarrow E_{D'}(\mathcal{T}) \wedge \bigwedge_{\beta \in J_{D'}} \Box\Diamond(\mathcal{T} \wedge \beta)$.

b) (COMPLETENESS) $Prov(\mathcal{T})$ is a Reiter extension of $\langle W, D\rangle$

Observe that the above theorem is a representation theorem for Reiter extensions through fixed points in $KD4Z$.

Example 3 Let $\langle W, D\rangle$ be $\langle \emptyset, \{\frac{:\beta}{\beta}\}\rangle$. Then $E(p) = \Box(\Diamond p \wedge (\Box\Diamond(p \wedge \beta) \rightarrow \Box(p \rightarrow \beta)))$.

$E(\top) = \Box(\Box\Diamond\beta \rightarrow \Box\beta)$, a minimal fixed point is $\Box\Box\beta$ which also verifies the stability condition since $\vdash_{KD4Z} \Psi(\Box\Box\beta)$ then $Prov(\mathcal{T}) = \{\beta\}$. On the other hand the other minimal fixed point $\Diamond\Box\neg\beta$ does not satisfy $\vdash_{KD4Z} \Psi(\Diamond\Box\neg\beta)$ $(=_{KD4Z} \Diamond\Box\neg\beta \rightarrow \Box(\Diamond\Box\neg\beta \wedge \Diamond\beta \leftrightarrow \bot) =_{KD4Z} \Diamond\Box\neg\beta \rightarrow \Box\Box\neg\beta$ therefore, in this case, the conditions for $Prov(\mathcal{T})$ to be a Reiter extension are not satisfied.

The above two theorems, together with the translation, show that we can effectively treat the $\Box$ as a nonmonotonic provability operator, where the nonmonotonic rule added to the modal language consists in recursively substituting any formula of the form $\Box\varphi$ with $\Box(C \rightarrow \varphi)$. Although we have not exhibited a modular translation of default into autoepistemic logic, which cannot be given [Got93], we have exhibited a *uniform embedding in the same monotonic logic* $KD4Z$. Now, it would be interesting to investigate how the lattice of fixed points is preserved in any logic Λ such that $KD4Z \leq \Lambda$.

5 Contextual expansions

Following the same contextual reasoning presented in the foregoing section we add to the modal logic Λ a nonmonotonic rule as follows. Let u and v be two new propositional variables treated as parameters:

$$\frac{E(u,v), \alpha \in \mathcal{T}, \beta \notin \mathcal{T}}{E(\alpha,\beta)} \qquad (18)$$

The meaning of (18) is that the substitution of the parameter u for any formula in $\mathcal{T}$ plays the role of *positive introspection* and the substitution of the parameter v for any formula not in $\mathcal{T}$ plays the role of *negative introspection*, for example the NBF rule. This rule generalizes the standard notion of nonmonotonic rule. We now look for suitable expansion like $E(\mathcal{T},\overline{\mathcal{T}}) = \mathcal{T}$, up to logical equivalence. For example, from the theory $\Box\alpha \rightarrow \alpha$ we can obtain $E(u,v) = \Box(u \rightarrow \alpha) \rightarrow \alpha$; here $\Box\alpha$ is interpreted as "α is provable from the context u", i.e. $\Box(u \rightarrow \alpha)$. The convention is that we may substitute any formula belonging to $\mathcal{T}$ for the parameter u. E.g. let $\mathcal{T}$ be a candidate for a fixed point and $\alpha \in \mathcal{T}$. We then get from $E(u,v)$ above, by substituting α for u, the formula $\Box(\alpha \rightarrow \alpha) \rightarrow \alpha$. If necessitation holds we get α hence all $\Box^n\alpha$ that is $Cn_\Lambda(\{\Box^n\alpha\}_{n<\omega})$ is a fixed point. Consider, now, the rules introduced in Section 3; if we add NBF, i.e. $\neg\Box v$, then $\mathcal{T}$ is the stable set of α. If Λ is an extension of $KD4$ and we use $E(u,v) = \neg\Box v \wedge \Box(\Box u \rightarrow u) \rightarrow \Box u$ or $E(u,v) = \neg\Box v \wedge \Box u$, with a set I of assumptions not containing u and v we get the autoepistemic expansions of I [ACP96b]. On the other hand if we add the $Eq^{\mathcal{T}}$, that is, $E(u,v) = \neg\Box v \wedge (\Box u \leftrightarrow u)$ (see [Sch95]) we get the nonmnotonic modal logic of Sw5.

We now give a constructive algorithm for finding fixed points for a particular syntactical form of $E(u,v)$ in the presence of NBF, when Λ is any modal logic extending $K4$. Algorithms to compute AEL expansions can be found, among the others in [MT91a, MT91b, MT93, Nie90, Nie92]. We shall write in the sequel $A \equiv_{\psi_1,\ldots\psi_n} B$, to mean that, with $\psi_0,\ldots\psi_n \notin \mathcal{T}$, $\neg\Box\psi_0 \wedge \ldots \wedge \neg\Box\psi_n$ are put in conjunction both with A and B in order to get the equivalence.

Theorem 8 Let $K4 \subseteq \Lambda$ and $\mathcal{T}$ stable and consistent. Let $E(u,v) = \neg\Box v \wedge B(\Box(u \rightarrow C_1(u)), \ldots, \Box(u \rightarrow C_m(u)))$, B non modal and such that each C is either of the above form or modal free not containing u. Then there are modal free formulae $\overline{B}$, $\psi_0\ldots\psi_m \notin \mathcal{T}$, $\varphi_0\ldots\varphi_k \in \mathcal{T}$ not containing u such that

$$E(u \wedge \overline{B} \wedge \varphi_0 \wedge \ldots \wedge \varphi_n, v) \equiv_{\psi_0,\ldots\psi_n} \overline{B}$$

Moreover if $\varphi_0 \wedge \ldots \wedge \varphi_k$ derives from $\overline{B}$ then $St(\overline{B}) = \mathcal{T}$ is an expansion.

Proof. Let us first consider $E(u) = B(\Box(u \rightarrow C_1),\ldots,\Box(u \rightarrow C_k))$, $k \leq n$, and B any modal formula with C_i the in most modal free subformulae not containing u. Let us construct the formula $\varphi_0 = \bigwedge_i C_i$ from all the $C_i \in \mathcal{T}$ and the conjunction of $E(u)$ with all $\neg\Box C_i$ such that $C_i \notin \mathcal{T}$. Then $\Box(u \wedge \varphi_0 \rightarrow C_i) = \Box\top$ iff $C_i \in \mathcal{T}$. If $C_i \notin \mathcal{T}$ then $\neg\Box C_i \in \mathcal{T}$ and $u \wedge \varphi_0 \rightarrow C_i \notin \mathcal{T}$ for all substitution for u which are in $\mathcal{T}$. Hence, by NBF, we get $\neg\Box(u \wedge \varphi_0 \rightarrow C_i)$, i.e. $\Box(u \wedge \varphi_0 \rightarrow C_i) \leftrightarrow \bot$. This means that $E(u) =_{C_i \notin \mathcal{T}} B(\Box(u \wedge \varphi_0 \rightarrow C_1),\ldots,\Box(u \wedge \varphi_0 \rightarrow C_m))$ can be reduced to a formula of degree DEGREE(E)-1 by substituting $\top$ or $\bot$ and using the First Substitution Lemma (see [Smo85, ACP96a]). Iterating the above reasoning we get modal free formulae $\overline{B}$, $\psi_1\ldots\psi_m \notin \mathcal{T}$ and $\varphi_0 \wedge \ldots \wedge \varphi_k$ not containing u such that

$$E(u \wedge \overline{B} \wedge \varphi_0\ldots\varphi_k) \equiv_{\psi_1,\ldots\psi_n} \overline{B} \qquad (19)$$

The claims follows because we may substitute any formula of $\mathcal{T}$ for u so that we can make the substitution $u \wedge \overline{B}$. It is an expansion if $\varphi_0 \wedge \ldots \wedge \varphi_k$ derives from $\overline{B}$.

Corollary 3 Let $E(u,v)$ be as in Theorem 8. Then $E(\mathcal{T},\overline{\mathcal{T}}) = \mathcal{T}$ iff $\mathcal{T}$ is an autoepistemic expansion of $E(\top)$.

Proof. Observe that we can use the previous proof by using the rule $\Box u$ with the set of assumptions $E(\top)$.

Example 4 Let $E(u) = \Box(u \rightarrow p) \rightarrow p$. Then $B(q)$ is $q \rightarrow p$. If we want to prove that the in most formula p is in $\mathcal{T}$, then we must put $\varphi_0 = p$. $\Box(u \wedge p \rightarrow p) = \top$ implies $E(\varphi_0 \wedge u) = p$. The algorithm stops because $E(\varphi_0 \wedge u)$ is not modal, that is, $\overline{B} = p$. We get the expansion $St(\{p\})$, since $\overline{B} = \varphi_0$ trivially holds. If $p \notin \mathcal{T}$ then let $\psi_1 = p$ and $\overline{B} = \bot \rightarrow p = \top$, in this case $St(\top)$ is an expansion too.

Example 5 Let $E(u) = p \rightarrow \Box(u \rightarrow p)$. $B(q)$ is $p \rightarrow q$. If the in most p, verifies $p \in \mathcal{T}$ then $\varphi_0 = p$ and $p \rightarrow \Box(u \wedge p \rightarrow p) = \top = \overline{B}$, that is $E(\varphi_0 \wedge u) = \top$. $\top = \overline{B}$ and we do not get the expansion $St(\{p\})$, because we do not derive p from $\overline{B}$. If $p \notin \mathcal{T}$ then let $\psi_1 = p$ and $\overline{B} = p \rightarrow \bot = \neg p$, in this case $St(\neg p)$ is an expansion.

Differently from the interpretation of $\Box$, given in the foregoing discussion, with this paradigm we do not require a specific meaning for $\Box$ as "it is provable", or at least there is no logical requirement for it. Therefore we can use the paradigm for embedding autoepistemic logic in any nonmonotonic modal logic in a range of modal logics containing $K4$ and defined by the MD&D construction. Suppose, for example, we want to gather into a single theory two different formalisms, e.g. defaults, in the Konolige translation, such as $\langle p : /q \rangle$ and $\langle s : /s \rangle$, and the autoepistemic clause $\Box p \rightarrow p$. Then let us translate the miscellaneous theory by maintaining for defaults the Konolige translation, i.e. $\Box p \rightarrow q$ and $\Box s \rightarrow s$ and for the autoepistemic the contextual one, i.e. $\Box(u \rightarrow p) \rightarrow p$. We get the stable set $St(\{p, q\})$ as solution if we use $S4f$ as underlying monotonic modal logic.

Acknowledgments

We thank Luigia Carlucci Aiello for having contributed to this research and Dov Gabbay for many useful discussions. Work carried out in the framework of the agreement between the Italian PT Administration and the Fondazione Ugo Bordoni.

References

[ACGP96] G. Amati, L. Carlucci Aiello, D. Gabbay, and F. Pirri. A structural property on modal frames characterizing default logic. *Journal of IGPL*, 4(1):1–24, 1996.

[ACP96a] G. Amati, L. Carlucci Aiello, and F. Pirri. *Definability and commonsense reasoning.* Third Symposium on Logical Formalization of Commonsense Reasoning, Stanford, USA, 1996.

[ACP96b] G. Amati, L. Carlucci Aiello, and F. Pirri. *Modal non monotonic reasoning via boxed fixed points.* 6th International Workshop on nonmonotonic reasoning, Oregon, USA, 1996.

[Ber75] C. Bernardi. The fixed-point theorem for diagonalizable algebras. *Studia Logica*, 34:239–251, 1975.

[Boo79] G. Boolos. *On the unprovability of consistency.* Oxford University Press, 1979.

[Bre91] G. Brewka. Cumulative default logic: in defence of nonmonotonic inference rules. *Artificial Intelligence Journal*, 50:183–206, 1991.

[Dix92] J. Dix. Default theories of Poole-type and a method for constructing cumulative versions of default-logic. In B. Neumann, editor, *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI-92)*, pages 289–293, 1992.

[Fit83] M. Fitting. *Proof Methods for Modal and Intuitionistic Logics.* Reidel Publishing Company, Dordrecht, 1983.

[FM94] C. Froidevaux and J. Mengin. Default logics: a unified view. *Computational Intelligence*, 10:331–369, 1994.

[GL88] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings of the Fifth International Conference on Logic Programming (ICLP-88)*, pages 230–237, 1988.

[GM94] L. Giordano and A. Martelli. On cumulative default reasoning. *Artificial Intelligence Journal*, 66:161–180, 1994.

[Gol87] R. Goldblatt. *Logics of time and computation.* CSLI, 1987.

[Got93] G. Gottlob. The power of beliefs or translating default logic into standard autoepistemic logic. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 570–575, 1993.

[Kon88] K. Konolige. On the relation between default and autoepistemic logic. *Artificial Intelligence Journal*, 35:343–382, 1988.

[Kon94] K. Konolige. Autoepistemic logic. In C.J.Hogger D.M.Gabbay and J.A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, pages 217–295. Oxford Science Publications, 1994.

[Lif85] Vladimir Lifschitz. Computing circumscription. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, pages 121–127, 1985.

[Luk88] W. Lukaszewics. Considerations on default logic: an alternative approach. *Computational Intelligence*, 4:1–16, 1988.

[McC80] J. McCarthy. Circumscription – a form of nonmonotonic reasoning. *Artificial Intelligence Journal*, 13:27–39, 1980.

[MD80] D. McDermott and J. Doyle. Nonmonotonic logic I. *Artificial Intelligence Journal*, 13:41–72, 1980.

[Moo85] R.C. Moore. Semantical considerations on nonmonotonic logics. *Artificial Intelligence Journal*, 25:75–94, 1985.

[Moo87] R.C. Moore. Possible-world semantics for autoepistemic logic. In M.L. Ginsberg, editor, *Readings in nonmonotonic reasoning*, pages 137–142. Morgan Kaufmann, 1987.

[MST93] V. W. Marek, G.F. Schwarz, and M. Truszczynski. Modal nonmonotonic logics: ranges, characterization, computation. *Journal for Association of Computing Machinery*, 40:963–990, 1993.

[MT91a] W. Marek and M. Truszczyński. Autoepistemic logic. *Journal of the ACM*, 38(3):588–619, 1991.

[MT91b] W. Marek and M. Truszczyński. Computing intersection of autoepistemic expansions. In *Proceedings of the First International Workshop on Logic Programming and Non Monotonic Reasoning*, pages 37–50. The MIT Press, 1991.

[MT93] V. W. Marek and M. Truszczynski. *Nonmonotonic Logic, context-dependent reasoning*. Springer-Verlag, 1993.

[Nie90] I. Niemelä. Towards automatic autoepistemic reasoning. In *Proceedings of the European Workshop on Logics in AI (JELIA-90)*, pages 428–443, 1990.

[Nie92] I. Niemelä. On the decidability and complexity of autoepistemic reasoning. *Fundamenta Informaticae*, 17(1,2):117–156, 1992.

[Rei80] R. Reiter. A logic for default reasoning. *Artificial Intelligence Journal*, 13:81–132, 1980.

[Sam75] G. Sambin. Un estensione del teorema di löb. *Rendiconti del Seminario Matematico dell'Università di Padova*, 52:193–199, 1975.

[Sch95] G.F. Schwarz. In search of a "true" logic of knowledge: the nonmonotonic perspective. *Artificial Intelligence Journal*, 79:39–63, 1995.

[Smo85] C. Smorynski. *Self-Reference and Modal Logic*. Springer-Verlag, Berlin, 1985.

[Sol76] R.M. Solovay. Provability interpretations of modal logic. *Israel Journal Math.*, 25:287–304, 1976.

[Tru91] M Truszczynski. Modal interpretations of default logic. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 393–398, 1991.

Nonmonotonic Logics and Logic Programming

An Argumentation-theoretic Approach to Reasoning with Specificity

Phan Minh Dung
Division of Computer Science,
Asian Institute of Technology
PO Box 2754, Bangkok 10501, Thailand
Email: dung@cs.ait.ac.th

Tran Cao Son
Department of Computer Science,
University of Texas at El Paso
El Paso, Texas 79968, USA
Email: tson@cs.utep.edu

Abstract

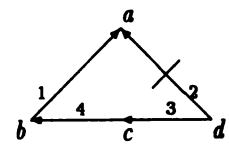
We present a new argumentation-theoretic approach to default reasoning with specificity. The new approach differs from other approaches in the way priority between defaults is handled. Here, it is context sensitive rather than context independent as in other approaches. We start by showing that any context independent handling of priorities between defaults as advocated in the literature until now is not sufficient to capture general defeasible inheritance reasoning. This motivates the introduction of an argumentation framework for default reasoning with specificity where the context sensitive priorities between defaults are captured by the attacks-relation between the arguments. We present several new and novel results. First we show that our framework subsumes the semantics of defeasible inheritance networks. We then show that the new semantics satisfies core properties of default reasoning such as the conditioning, deduction, reduction, and cumulative propositions [1]. To give a proof procedure to our approach, we present a modular transformation from default theories into logic programs where the preferred semantics of the former coincides with the answer set semantics of the later.

1 Introduction

Default reasoning has emerged as a response to the need of a mechanism that can accommodate the nonmonotonic character of commonsense reasoning [17, 18, 20, 29]. In default reasoning, default rules are used to draw new conclusions in the absence of conflicting arguments. One of the most difficult problems in default reasoning is the problem of reasoning with specificity. Though many proposals have been forwarded in the literature [1, 3, 11, 12, 26, 30] to address it, no satisfactory solution is obtained yet.

Formally a default theory T could be defined as a pair (E, K) where E is a set of evidence or facts representing what we call the concrete context of T, $K = (D, B)$ is the domain knowledge consisting of a set of default rules D, and a first order theory B representing the background knowledge. In the literature [2, 5, 11, 12] the principle of reasoning with specificity is "enforced" by first determining a set of priority orders between defaults in D using the information given by the domain knowledge K. Based on these priorities between defaults and following some sensible and intuitive criteria [1, 3, 11, 12], the semantics of T is then defined either model-theoretically by selecting a subset of the set of all models of $E \cup B$ as the set of preferred models of T or proof-theoretically be selecting certain extensions as the preferred extension [5, 30]. The problem of these approaches is that their obtained semantics is rather weak. We will show in section 2 that these approaches do not capture general defeasible inheritance reasoning. The reason is that the priorities between defaults are defined independent of the context E.

It seems that there is a consensus in the literature [5, 11, 12] recognizing that for any default reasoning system to account for general inheritance reasoning it is necessary to combine the three important ideas of minimality, conditioning and anticontrapositive. The following example illustrates this point.



Example 1.1 Let consider the default theory $T = (E, K)$ with $B = \emptyset$, $D = \{d/c, c/b, d/\neg a, b/a\}$, and

[1] To the best of our knowledge, it is the first time a general framework for default reasoning has been given which subsumes inheritance reasoning when applied to nonmonotonic inheritance networks and at the same time also satisfies the core properties of default reasoning

$E = \{d\}$ represented in the above picture. The idea of minimality restricts the set of candidates for preferred models to those in which only a minimal set of default rules are violated (or equivalently a maximal set of default rules are satisfied). In this case, the "minimal" models are:

$M_1 = \{d, c, b, \neg a\}$ with $D_1 = \{2, 3, 4\}$ being the set of rules satisfied in M_1.

$M_2 = \{d, c, b, a\}, D_2 = \{1, 3, 4\}$

$M_3 = \{d, \neg c, \neg b, \neg a\}, D_3 = \{1, 2, 4\}$

$M_4 = \{d, c, \neg b, \neg a\}, D_4 = \{1, 2, 3\}$

where in the last two models, some rules are "applied" contrapositively. The "anti-contrapositive"-idea forbids contrapositive applications of default rules. Hence, the set of candidates for preferred models in the above example would reduce to $\{M_1, M_2\}$. Finally, the idea of conditioning stating that given an evidence p in a default theory (E, K) containing a default rule p/q, q should be concluded, will pick M_1 as the preferred model.

The idea of minimality is the basic idea of circumscription [18]. To account for the other two, prioritized circumscription has been introduced with some success (see, e.g. [22]). Reiter's default logic [29] can be viewed as a first attempt to combine proof theoretically the idea of minimality and anticontrapositive. Seminormal defaults and other more recent proposals [5, 11, 12, 30] have been proposed to incorporate the idea of conditioning into default logic. Geffner and Pearl [11] showed that combining the ideas of minimality and conditioning can provide a system which can account for reasoning with specificity and a weak form of inheritance reasoning.

Argumentation has been recognized lately as an important and natural approach to nonmonotonic reasoning [4, 6, 7, 11, 21, 25, 28, 30, 37]. Dung [6], Bondarenko et all. [4] showed that most of the major nonmonotonic logics [17, 18, 20, 23, 29] are special cases of a general simple argumentation system. Geffner and Pearl [11] have used argumentation to give a proof procedure for their conditional logic. In this paper, we use argumentation to combine the three basic ideas of default reasoning where a sort of context dependent priorities between defaults is (implicitly) employed. We then demonstrate that our new framework can capture general defeasible inheritance reasoning by showing that the credulous semantics of defeasible inheritance networks [13, 14, 15, 19, 32, 33, 34] are captured in our framework. We also show that our semantics satisfies core properties of default reasoning such as the conditioning, deduction, reduction, and cumulative propositions.

To give proof procedures to our semantics we transform default theories into logic programs such that the answer set semantics of the later coincides with the argumentational semantics of the former. So, well-known proof procedures for logic programs can be used as proof-procedure for default reasoning.

The paper is organized as follows. In section 2 we will show that any default reasoning system based on a context independent priorities between defaults can not account in full for general inheritance reasoning. In section 3 we recall the argumentation framework of Dung [6]. Section 4 presents our new approach, where we show that our framework when applied to inheritance reasoning subsumes the credulous semantics of defeasible inheritance networks. We also show that the new semantics enjoy many important properties of an entailment relation discussed in [11, 16] such as conditioning, deductive, reduction, and cumulative proposition. Afterward we present the transformation of default theories into extended logic programs where the preferred semantics of the former coincides with the answer set semantics of the later. We conclude in section 6.

2 What is Wrong with Context Independent Priorities

Let $T = (E, K)$ be a default theory where E (a first order theory) is called the context of T representing the evidence or facts, $K = (B, D)$ is called the domain knowledge of T with B (a first order theory) representing the background knowledge and D being a set of defaults. In this section, we will give an example showing that priority-based approaches to default reasoning, in which priorities between defaults are independent of the context E, do not capture general default reasoning.

In the literature [1, 3, 5, 11, 12, 24, 30], the semantics of T is defined based on certain partial orders on D which is determined solely by K. Let PO_K be the set of all these partial orders. For each partial order $\alpha \in PO_K$, where $(d, d') \in \alpha$ means that d is of lower priority than d', a partial order $<_\alpha$ between the sets of defaults in D is defined where $S <_\alpha S'$ means that S is preferred to S'. There are many ways to define $<_\alpha$ [1, 3, 5, 11, 12]. But whatever the definition of $<_\alpha$ is, $<_\alpha$ has to satisfy the following property.

Let S be a subset of D and d, d' be two defaults in D such that $(d, d') \in \alpha$. Then $S \cup d' <_\alpha S \cup d$.

$<_\alpha$ can be extended into an partial order between models of $B \cup E$ as follows:

$M <_\alpha M'$ iff $D_M <_\alpha D_{M'}$

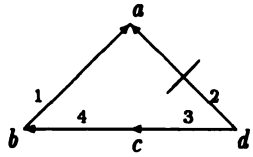
where D_M is the set of all defaults in D which are satisfiable in M and a default p/q is said to be satisfiable in M iff the material implication $p \rightarrow q$ is satisfiable in M.

A model M of $B \cup E$ is defined as a preferred model of T if there exists a partial order α in PO_K such that M is minimal with respect to $<_\alpha$. We then say that a formula β is defeasibly derived from T if β holds in each preferred model of T.

Now we want to show that any preferential semantics based on $<_\alpha$ can not account in full for general inheritance reasoning.

Let us consider again the example:

Example 2.1 Let B be empty, $E = \{d\}$, and D be:



An interpretation for the above network can be given by *d=student, c=young adult, b=adult, a=married.* The desirable semantics here is represented by the model $M = \{d, c, b, \neg a\}$. To have this semantics, all priority-based approaches in the literature [1, 3, 5, 12] assigns the default 1 a lower priority than the default 2. Note that though conditional entailment [11] does not provide the desirable semantics in this case, it could be "fixed" by restricting the set of partial orders PO_K on those in which 1 has a lower priority than 2. Further, it is important to note that the priorities between defaults are determined independent of the context E.

Let consider T under a new context $E' = \{d, \neg c, b\}$. Now, since c does not hold, the default $d/\neg a$ can not be considered more specific than the default b/a. Hence, it is intuitive to expect that neither a nor $\neg a$ should be concluded in this case. This is also the result sanctioned by all semantics of defeasible inheritance networks [13, 14, 15, 19, 32]. In any priority-based system employing the same priorities between defaults with respect to E' as with respect to E, we have $M = \{\neg a, d, \neg c, b\} <_\alpha M' = \{a, d, \neg c, b\}$ since $D_M = \{4, 2\} <_\alpha D'_M = \{4, 1\}$ (due to $(1, 2) \in \alpha$). That means priority-based approaches in the literature would conclude $\neg a$ given (E', K) which is certainly not the intuitive result we expect. This leads to the idea that 1 should have lower priority than 2 only if the context $\{d, c, b\}$ holds. In other words, the priority order under the context $\{d, \neg c, b\}$ is different than the priority order under the context $\{d, c, b\}$. It turns out that this context-sensitive priority can be naturally captured in an argumentation theoretic approach.

3 Preliminaries: Argumentation Framework

We first recall the basics of the abstract theory of argumentation of Dung [6].

Definition 3.1 *An* argumentation framework *is a pair* $AF = \langle AR, attacks \rangle$, *where* AR *is a set of arguments, and* $attacks \subseteq AR \times AR$.

If $(A, B) \in attacks$ we say A attacks B or B is attacked by A. Further a set of arguments S is said to attack an argument A if some argument in S attacks A.

Definition 3.2 *A set of arguments S is said to be* conflict-free *if there exist no arguments A,B in S such that* $(A, B) \in attacks$.

The stable semantics of AF is defined as follows.

Definition 3.3 *A set of arguments S is called a* stable extension *of AF if S is conflict-free and S attacks every argument which does not belong to S.*

It is easy to see that S is stable iff S={A | A is not attacked by S}.

The stable semantics of argumentation framework captures the semantics of many other mainstream approaches to nonmonotonic reasoning such as extension of Reiter's Default Logic [29], stable expansion of Autoepicstemic Logic [17], and stable model of Logic Programming [9]. Also, the credulous semantics of an inheritance network coincides with the stable semantics of the corresponding argumentation framework [7].

Often, a more skeptical semantics is advocated in many approaches to nonmonotonic reasoning [25, 36]. This form of skeptical semantics is captured in the argumentation framework by the notion of grounded extension defined as the least fixpoint of the following operator.

$$F_{AF} : 2^{AR} \rightarrow 2^{AR}$$

$F_{AF}(S) = \{A \mid$ A is defended by S $\}$ where A is *defended by S* iff for every argument B, if B attacks A, then S attacks B.

The *grounded semantics* of an argumentation framework AF is defined as the least fixpoint of F_{AF}. It has been pointed out in [6] that both the semantics of Pollock's Inductive Defeasible Logic [25], and the well-founded semantics of Logic Programming [36] are captured by the grounded semantics of argumentation.

The preferred semantics of argumentation is defined as follows

Definition 3.4 *A set of arguments S is called a* preferred extension *of AF iff S is a conflict-free maximal fixpoint of* F_{AF}.

In general, stable extensions are preferred extensions but not vice versa. But as we will see later, in the argumentation frameworks corresponding to default theories, stable semantics and preferred semantics coincide. So it is enough for us to work with either of them.

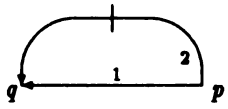
4 A New Approach

In this section we present a new approach to handle specificity in a class of default theories which comprises of a set of ground literals (evidences) E and a finite set of default rules $D = \{d_1, \ldots, d_n\}$ where each rule $d \in D$ is of the form $l_1, \ldots, l_m/h$ and $l_i's$, h are literals over an arbitrary but fixed first order language $\mathcal{L}$ such that $\{h\} \cup \{l_1, \ldots, l_m\}$ is consistent. h is called the *head* of d, denoted by $hd(d)$, and the set $\{l_1, \ldots, l_n\}$ is called the *body* of d, denoted by $bd(d)$. Two defaults d and d' are said to be *in conflict* if $\{hd(d), hd(d')\}$ is inconsistent. Throughout this paper, we will assume that the set of background knowledge in T is empty, i.e. $B = \emptyset$. The restriction that B is empty is taken to allow us to present the main ideas of our new approach in an intuitive and simple way without bothering to much with technical details. We also assume that the language $\mathcal{L}$ is finite, i.e., the sets of constant, function, and predicate symbols of $\mathcal{L}$ are finite. The set of all ground instances of the defaults in D is denoted by *ground(D)*.

Definition 4.1 *1. A default theory $T = (E, D)$ is* inconsistent *if E is inconsistent or there exists two defaults d and d' in D such that $bd(d) \equiv bd(d')$ and $\{hd(d), hd(d')\}$ is inconsistent.*

2. A default theory is consistent *if it is not inconsistent.*

An example of an inconsistent default theory is illustrated in the following picture.



We now define the *consequence* relation.

Definition 4.2 *Given an evidence set E and a default set D we say that a ground literal l is a* consequence of E wrt. D, *denoted by $E \vdash_D l$, iff $l \in E$ or there is a sequence of ground defaults $d_1, \ldots, d_n$ $(1 \leq n)$ in* ground(D) *such that*

1. $l = hd(d_n)$ *and*
2. $bd(d_1) \subseteq E$, *and*
3. $bd(d_{i+1}) \subseteq E \cup \{hd(d_1), \ldots, hd(d_i)\}$.

For a set of literals L we write $E \vdash_D L$ iff $\forall l \in L: E \vdash_D l$.

We write $E \vdash_D false$ iff there is a ground atom a such that both $E \vdash_D a$ and $E \vdash_D \neg a$ hold.

We next introduce the notion of *atom dependency graph* of a default theory $T = (E, D)$ before we define the class of acyclic default theories.

Definition 4.3 *Let $T = (E, D)$ be a default theory. The* atom dependency graph *of T is defined as a directed graph G_T where the nodes of G_T are ground atoms in the language $\mathcal{L}$ of T and there is a directed link from b to c in G_T iff there is a default $b_1, \ldots, b_n/h$ in ground(D) such that b is the atom of some b_i and c is the atom of h.*

Definition 4.4 *A default theory $T = (E, D)$ is* acyclic *if there is no cycle in the atom dependency graph of T.*

In following, we will restrict our study only on acyclic and consistent default theories. We believe that the class of acyclic and consistent default theories is large enough to accommodate the most of practical default reasoning. At least, most of the examples found in the literature belong to this class.

From now on, $T = (E, D)$ denotes an arbitrary but fixed acyclic and consistent default theory if not otherwise specified.

Definition 4.5 *A set of ground defaults $A \subseteq ground(D)$ is called an* argument *in T if for all $d \in A$, $E \vdash_A bd(d)$, and $E \not\vdash_A false$.*

By definition, in the example 2.1 $A_1 = \{2, 3, 4\}$, $A_2 = \{1, 4, 3\}$, $A_3 = \{2\}$ are arguments while $B = \{1, 2\}$ is not an argument since $\{d\} \not\vdash_B bd(1)$.

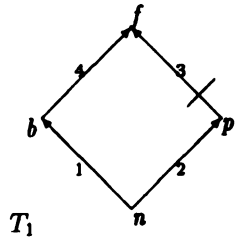
For any default theory T, the set of all arguments in T is denoted by AR_T.

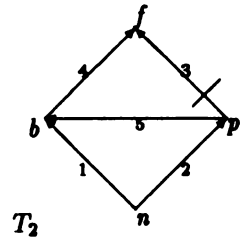
Let A be an argument and w be a ground literal, we say A *supports* w iff $E \vdash_A w$. Further $Con(A)$ denotes the set of all conclusions supported by A, i.e., $Con(A) = \{w | E \vdash_A w\}$. It is easy to see that if A and B are two arguments with consistent $Con(A) \cup Con(B)$ then $A \cup B$ is an argument.

To define the semantics of T, we now determine the attack- relationship between the arguments in AR_T. First, it is easy to see that A *attacks* B if A and B support conflicting conclusions, i.e., $Con(A) \cup Con(B)$ is inconsistent.

We now consider another kind of attack.

Example 4.1 (Motivation of Attack by Specificity)





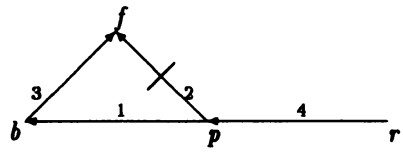
Consider the default theory T_1 in the above figure with $E = \{n\}$. Let $A_1 = \{2, 3\}$ and $A_2 = \{1, 4\}$. From the above discussion, it is clear that A_1 attacks A_2 and

vice versa. Now adding the default p/b to T_1 we will have the famous Penguin-Bird-Fly example in which A_1 becomes more specific than A_2. Thus, due to the specificity principle which stipulates that specific information overrides more general but conflicting one we can say that adding p/b to T_1 creates an attack of new kind against $A_2 = \{1,4\}$. Hence it is natural to view the argument $B = \{2,5\}$ in the presence of the default 3 as an attack against the argument A_2. □

The above example shows that there are two types of attacks among the arguments in AR_T: attacks by conflict and attack by specificity. The definition of the first type of attacks is clear. It remains to define the second type of attacks. It is not difficult to see that in the above example, the attack by specificity is determined by the context consisting of three defaults 3,4,5 in which default 5 makes default 4 less specific than default 3. Such contexts are called specificity-relevant minimal conflict sets. A precise definition is given in definition 4.10. But first, we define the notion of *minimal conflict sets.*

Definition 4.6 *Let d be a default in ground(D). A set of ground defaults $C \subseteq$ ground(D) is called a* minimal conflict set (abbr. **MCS**) wrt. d, *iff*

- $d \in C$, *and*
- C *is a minimal set such that* $bd(d) \vdash_C$ *false, and*
- *there exists no* $d' \in C$ *such that* $bd(d) \not\equiv bd(d')$ *and* $bd(d') \vdash_C$ *false*



Example 4.2 In the above picture, $C = \{1,2,3\}$ is a MCS wrt. 2 or 1 but $C' = \{1,2,3,4\}$ is not a MCS wrt. 4 since the third condition is violated. □

The following lemma follows directly from definition of MCS.

Lemma 4.1 *Let C be an arbitrary MCS wrt. d.*

- *For all* $d' \in C$ *there exists* $C' \subseteq C$, *such that* $bd(d) \vdash_{C'} bd(d')$.
- C *contains exactly two defaults* d' *and* d'' *such that* $\{hd(d'), hd(d'')\}$ *is inconsistent.*

Definition 4.7 *Let C be a MCS. A default $d' \in C$ is called a* conflicted default *in C if* $\exists d'' \in C$ *such that* $\{hd(d'), hd(d'')\}$ *is inconsistent.*

To resolve the conflict caused by a MCS we will block one of its defaults from applying and to enforce the principle of specificity the least specific default should be blocked as suggested in [5]. For example, in T_2 of example 4.1 we should block 4 rather than 3.

Definition 4.8 *Let C be a MCS and* $d', d'' \in C$. d'' *is said to be* more specific *than d' (with respect to C) iff there exists* $C' \subseteq C$ *such that* $bd(d'') \vdash_{C'} bd(d')$.

Obviously, the more specific relation is a pre-order with respect to a MCS, i.e., in a MCS it is transitive and reflexive.

Definition 4.9 *Let C be a MCS and* $d' \in C$. d' *is said to be* most specific *in C iff for all defaults* $d'' \in C$, d' *is more specific than* d''.

Let C be a MCS with respect to d in an acyclic default theory. Then, it is easy to see that $d' \in C$ is most specific iff $bd(d) \equiv bd(d')$.

In example 2.1, the defaults 1 and 2 are the two conflicted defaults of the MCS of which 2 is a most specific default. In T_1 of example 4.1 we have two conflicted defaults 3 and 4 but none of them is most specific. We distinguish these two types of MCS's in the following definition.

Definition 4.10 *A MCS C is said to be* specificity relevant *iff one of the two conflicted defaults of C is most specific in C.*

By definition, the MCS's of example 2.1 are specificity relevant while the MCS's in T_1 of example 4.1 are not. The following easy lemma helps to characterize the specificity relevant MCS's.

Lemma 4.2 *Let C be a MCS with respect to d. Then C is specificity relevant if d is a conflicted default.*

For convenience, from now on, whenever we say that C_d is a specificity relevant MCS, we understand that d is one of the conflicted defaults.

We have specified a class of MCS's which are relevant for specificity reasoning. We now give a formal definition for the attack relationship.

Definition 4.11 *Let A, B be arguments in AR_T. We say that A attacks B if one of the following conditions holds:*

1. attack by conflict: *A and B supports conflicting conclusions, i.e., $Con(A) \cup Con(B)$ is inconsistent.*
2. attacks by specificity: *there exists a specificity relevant MCS C_d whose conflicted defaults are d, d' such that*

- $E \vdash_A bd(d)$, *and* $C_d \setminus \{d', d\} \subseteq A$, *and*
- $d' \in B$.

$AF_T = (AR_T, attacks)$ *denotes the argumentation framework corresponding to* T.

In example 4.1 the default theory T_1 has following arguments: $A_0 = \emptyset, A_1 = \{2,3\}, A_2 = \{1,4\}, A_3 = \{2\}, A_4 = \{1,2,3\}, A_5 = \{1\}, A_6 = \{1,2,4\}, A_7 = \{1,2\}$, and $attacks_{T_1} = \{(A_2, A_1), (A_1, A_2), (A_2, A_4), (A_4, A_2), (A_6, A_1), (A_1, A_6), (A_6, A_4), (A_4, A_6)\}$. We then have $\{A_0, A_3, A_5, A_1, A_4, A_7\}$ and $\{A_0, A_3, A_5, A_2, A_6, A_7\}$ are the two preferred extensions which are also stable, of the argumentation framework $(AR_{T_1}, attacks_{T_1})$ which is the expected result. Introducing p/b into T_1 creates some more arguments $A_8 = \{2,5\}$, $A_9 = \{2,5,4\}$, $A_{10} = \{1,2,5\}$, $A_{11} = \{1,2,4,5\}$, $A_{12} = \{1,2,5,3\}$, and $A_{13} = \{2,3,5\}$ and for all arguments of AF_{T_2} we have i) if $\{2,5\} \subseteq A, 4 \in B$ then A attacks B and ii) if $3 \in A, 4 \in B$ then A attacks B and B attacks A. It is easy to see that AF_{T_2} has only one preferred extension $\{A_0, A_1, A_3, A_4, A_5, A_7, A_8, A_{10}, A_{12}, A_{13}\}$ which is also stable. □

As stated in section 3, the coincidence of preferred semantics and stable semantics in AF_T is proved in the following theorem.

Theorem 1 *Let* T *be an acyclic and consistent default theory. Then, each preferred extension of* AF_T *is also a stable extension of* AF_T *and vice versa.*

Remark 4.1 *From now on, we often refer to a preferred extension of* AF_T *simply as a preferred extension of* T.

It has been shown in [6] that preferred extension always exists for argumentation systems. So, we have

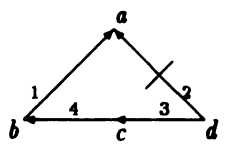
Lemma 4.3 *Each acyclic and consistent default theory has at least a preferred extension.*

Definition 4.12 *We write* $T \mid\sim w$ *for a ground literal* w *iff* w *is supported by at least one argument in each preferred extension of* T.

How Context-Sensitive Priorities Are Handled

The next example demonstrates that our approach deals with default priorities in a context sensitive way and hence provides the expected intuitive results in many cases where other approaches fail.

Example 4.3 (Continuation of example 2.1)



Consider $T = (E, D)$ with $D = \{1,2,3,4\}$. We consider two sets of evidence:

1. $E = \{d\}$. Then, $AR_T = \{A_0, A_1, A_2, A_3, A_4, A_5, A_6\}$ where $A_0 = \emptyset$, $A_1 = \{2\}$, $A_2 = \{3\}$, $A_3 = \{3,4\}$, $A_4 = \{1,3,4\}$, $A_5 = \{2,3,4\}$, and $A_6 = \{2,3\}$. $attacks_T = \{(A_1, A_4), (A_4, A_1), (A_4, A_5), (A_5, A_4), (A_3, A_4), (A_6, A_4), (A_4, A_6)\}$. We can say that 2 has higher priority than 1 in the context $\{d\}$ by the fact that A_4, the only argument using 1, is attacked by A_3 while A_3 does not attack the arguments containing 2. It is easy to see that AF_T has **only one preferred extension** $\{A_0, A_1, A_2, A_3, A_5, A_6\}$ which supports b, c, and $\neg a$ as in defeasible inheritance network. □
2. $E = \{d, \neg c, b\}$. Then, AF_T contains only three arguments: $A_0 = \emptyset$, $A_1 = \{2\}$, $B = \{1\}$. $attacks_T = \{(A_1, B), (B, A_1)\}$. From the symmetry in the attacks-relation between arguments A_1, B we can say that there are no priorities between defaults 1 and 2. AF_T has **two preferred extensions** $\{A_0, A_1\}$, and $\{A_0, B\}$ and so $T \not\mid\sim \neg f$ and $T \not\mid\sim f$. □

Core Properties of $\mid\sim$

We now prove some general propositions of $\mid\sim$. The next theorem shows that conditioning is satisfied in preferred semantics.

Theorem 2 *(Conditioning) Given an acyclic and consistent default theory* $T = (E, D)$ *with* $E = bd(d)$ *for an arbitrary ground default* $d \in ground(D)$. *Then,* $T \mid\sim hd(d)$.

We will next discuss other general propositions [1, 5, 11, 16, 17] of the entailment relation $\mid\sim$. The deduction proposition is trivial because $\emptyset$ is an argument which supports the evidence E and $\emptyset$ belongs to every preferred extension of AF_T as there is no argument which attacks $\emptyset$. The reduction and cumulative propositions of $\mid\sim$ are proved in the next theorems.

Theorem 3 *(Reduction) Given T=(E,D) such that* $T \mid\sim a$ *and* $T' \mid\sim b$ *for* $T' = (E \cup \{a\}, D)$. *Then,* $T \mid\sim b$.

Proof Let M be an arbitrary preferred extension of T. Lemma 6.3 (appendix 2) says that M is also a preferred extension of T'. Hence, b is supported in M. Therefore $T \mid\sim b$. The theorem is proved. □

Theorem 4 *(Cumulativity) Let* $T = (E, D)$ *be a default theory such that* $T \mid\sim a$ *and* $T \mid\sim b$. *Then,* $T' \mid\sim b$ *for* $T' = (E \cup \{a\}, D)$.

Proof Follows directly from lemma 6.7 and 6.8 (appendix 2). □

Coincidence between Preferred Extensions and Credulous Extensions of Defeasible Inheritance Network

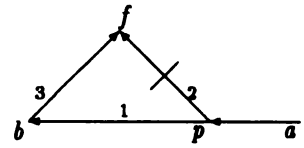
In this section we show that each defeasible inheritance network can be viewed as a default theory where the credulous semantics of the former coincides with the preferred semantics of the later. A defeasible inheritance network Γ is defined here as a finite collection of positive and negative direct links between nodes. If x, y are nodes then $x \rightarrow y$ (resp. $x \not\rightarrow y$) represents a positive (resp. negative) *direct link* from x to y. A network Γ is *consistent* if there exist no link $x \rightarrow y$ such that $x \not\rightarrow y$ also belongs to Γ. A *positive path* from x_1 to x_n through x_2, ..., x_{n-1}, denoted by $\pi(x_1, \sigma, x_n)$, is a sequence of direct links $x_1 \rightarrow x_2$, $x_2 \rightarrow x_3$, ... , $x_{n-1} \rightarrow x_n$. Similarly, a *negative path* from x_1 to x_n through x_2, ..., x_{n-1}, denoted by $\bar{\pi}(x_1, \sigma, x_n)$, is a sequence of direct links $x_1 \rightarrow x_2$, $x_2 \rightarrow x_3$, ... , $x_{n-1} \not\rightarrow x_n$. A *generalized path* is a sequence of direct links (x_1, x_2), (x_2, x_3), ... , (x_{n-1}, x_n), where (x_i, x_{i+1}) denotes a positive or negative direct link. Γ is *acyclic* if there is no generalized path (x_1, x_2), (x_2, x_3), ... , (x_{n-1}, x_n) with $x_1 = x_n$. Any path of the form $\pi(x, \sigma, y)$ *conflicts* with any path of the form $\bar{\pi}(x, \tau, y)$ and vice versa. Let Φ be a set of paths in Γ. A path σ is *conflicted in* Φ iff Φ contains a path that conflicts with σ. A positive path $\pi(x, \sigma, u) \rightarrow y$ is *constructible* in Φ iff $\pi(x, \sigma, u) \in \Phi$ and $u \rightarrow y \in \Gamma$. A negative path $\pi(x, \sigma, u) \not\rightarrow y$ is constructible in Φ iff $\pi(x, \sigma, u) \in \Phi$ and $u \not\rightarrow y \in \Gamma$. A positive path $\pi(x, \sigma, u) \rightarrow y$ is *preempted* in Φ iff there is a node v such that (i) $v \not\rightarrow y \in \Gamma$ and (ii) either $v = x$ or there is a path of the form $\pi(x, \tau_1, v, \tau_2, u) \in \Phi$. A negative path $\pi(x, \sigma, u) \not\rightarrow y$ is *preempted* in Φ iff there is a node v such that (i) $v \rightarrow y \in \Gamma$ and (ii) either $v = x$ or there is a path of the form $\pi(x, \tau_1, v, \tau_2, u) \in \Phi$. A *credulous extension* is a set Φ such that $\Phi = \{\alpha \mid \alpha$ is constructible, not conflicted, and not preempted in $\Phi\}$.

In the following Γ denotes a fixed but arbitrary defeasible inheritance network. A node x in Γ is called an *individual node* if there exists no direct link of the form $y \rightarrow x$ or $y \not\rightarrow x$ in Γ. Let I_Γ denote the set of individuals of Γ. An individual link is a link started from an individual node. A default link is a link which is not an individual link. The set of all default links in Γ is denoted by Γ_D. The inheritance network Γ can be represented as the default theory $T_\Gamma = (E, D)$ where

$D = \{p(x)/q(x) | p \rightarrow q \in \Gamma_D\} \cup \{p(x)/\neg q(x) | p \not\rightarrow q \in \Gamma_D\}$ and

$E = \{x(a) | a \rightarrow x \in \Gamma,\ a \in I_\Gamma\} \cup \{\neg x(a) | a \not\rightarrow x \in \Gamma,\ a \in I_\Gamma\}$

Example 4.4 Let Γ be an inheritance network represented by the picture below. It is easy to see that $T_\Gamma = (\{p(a)\}, \{p(x)/b(x), p(x)/\neg f(x), b(x)/f(x)\})$. □



A path in Γ is called *ground path* if it begins with an individual a.

Definition 4.13 *1. Let $p(a)$ be a ground literal. Then we write $\Gamma \mid\sim_c p(a)$ if each credulous extension of Γ contains a ground path of the form $a \rightarrow x_1 \rightarrow \ldots \rightarrow x_n \rightarrow p$*

2. Similarly we write $\Gamma \mid\sim_c \neg p(a)$ if each credulous extension of Γ contains a ground path of the form $a \rightarrow x_1 \rightarrow \ldots \rightarrow x_n \not\rightarrow p$.

The following theorem shows that our framework captures nonmonotonic inheritance reasoning.

Theorem 5 *Let Γ be an acyclic and consistent inheritance network and T_Γ be the corresponding default theory. Then for each ground literal w,*

$$\Gamma \mid\sim_c w \quad iff \quad T_\Gamma \mid\sim w$$

5 Transforming into Logic Programming

In this section we present a transformation from a default theory $T = (E, D)$ into an extended logic program P_T and prove the coincidence between the answer set semantics of P_T and the preferred extension semantics of T. We first recalled the notion of answer set of extended logic programs and then present the transformation from T to P_T. An *extended logic program* P [10] is a set of rules of the form

$$L_0 \leftarrow L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n,$$

where $0 \leq m \leq n$, each L_i is a literal, and *not* represents the negation-as-failure.

Let *Lit* denote the set of ground literals in the language of the program P.

1. If P does not contain *not* L (i.e. $m = n$ in every rule of P) an answer set of P is defined as the smallest set S, $S \subseteq Lit$ such that
   (a) for any ground instance $L_0 \leftarrow L_1, \ldots, L_m$ of a rule from P, if $L_1, \ldots, L_m \in S$, then $L_0 \in S$.
   (b) if S contains a pair of complementary literals, then $S = Lit$.
2. If the program P does contain *not* ($m < n$ in some rule of P), $S \subseteq Lit$ is an answer set of P if S is the answer set of the program P' obtained from the set of all ground instances of P by deleting

(a) each rule that has a formula *not* L in its body with $L \in S$, and

(b) all formulas of the form *not* L in the bodies of the remaining clauses.

Let $T = (E, D)$ be a default theory. To each default rule d_i: $l_1, \ldots, l_n / h_i$ we associate an unique abnormal predicate ab_i and write $hd_i(x)$ for $h_i(x)$, $bd_i(x)$ for $\{l_1(x), \ldots, l_n(x)\}$. Then, T is transformed into an extended logic program P_T as follows.

1. For each rule d_i, the two clauses

$$hd_i(x) \leftarrow bd_i(x),\ not\ ab_i(x)$$

$$ab_i(x) \leftarrow \neg hd_i(x)$$

belong to P_T, and

2. For each specificity relevant MCS C_{d_j} where d_j, d_t are its two conflicted defaults then the clause

$$ab_t(x) \leftarrow \bigwedge_{k \in C \setminus \{j,t\}} bd_k(x),\ not\ ab_k(x)$$

belongs to P_T, and

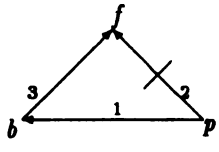
3. For each ground literal $l \in E$ the clause

$$l \leftarrow$$

is added to P_T.

We illustrate the transformation from T into P_T in the next examples.

Example 5.1 (Penguin)



P_T will contain following clauses:

$$\begin{array}{lcl} b(x) & \leftarrow & p(x), not\ ab_1(x) \\ ab_1(x) & \leftarrow & \neg b(x) \end{array}$$

$$\begin{array}{lcl} \neg f(x) & \leftarrow & p(x), not\ ab_2(x) \\ ab_2(x) & \leftarrow & f(x) \end{array}$$

$$\begin{array}{lcl} f(x) & \leftarrow & b(x), not\ ab_3(x) \\ ab_3(x) & \leftarrow & \neg f(x) \end{array}$$

$$ab_3(x) \leftarrow p(x), not\ ab_1(x)$$

Let $E = \{p(a)\}$. Then, P_T has one more clause: $p(a) \leftarrow$. In this case we have only one answer set: $\{p(a), b(a), \neg f(a), ab_3(a)\}$. □

Definition 5.1 *Let P be an arbitrary extended logic program and w be an arbitrary ground literal. We write $P \vdash_{ans} w$ iff w is contained in each answer set of P.*

Theorem 6 *Let T be an acyclic and consistent default theory. Then for each ground literal w in the language of T,*

$$T \vdash w \quad iff \quad P_T \vdash_{ans} w$$

From the construction of P_T we can easily prove the following theorem about the modularity of the transformation from T to P_T.

Theorem 7 *Given a $T = (E, D)$ and $T' = (E', D')$ with $E \subseteq E'$ and $D \subseteq D'$. Then, $P_T \subseteq P_{T'}$.*

It is easy to see that if we translate the extended logic program P_T into a general logic program P'_T by: i) introducing a new literal $a'(x)$ representing the negative literal $\neg a(x)$ for each atom $a(x)$ in the language of P_T, and ii) replacing each negative literals $\neg a(x)$ of P_T by the newly introduced literal $a'(x)$ (see [10] for more detail about this transformation) then the stable semantics of the normal logic program P'_T coincides with the answer set semantics of P_T.

Theorem 8 *1. Let M be a stable model of P'_T. Then the set of literals obtained by replacing each atom of the form $a'(t)$ in M by $\neg a(t)$ is an answer set of P_T.*

2. Let M be an answer set of P_T. Then the set of atoms obtained by replacing each literals of the form $\neg a(t)$ in M by $a'(t)$ is a stable model of P'_T.

6 Conclusions

A new approach for reasoning with specificity in acyclic and consistent default theories has been proposed where the three basic ideas - "maximality", "anti-contrapositive", and "specific overriding" - have been combined to solve the specificity problem naturally. Here it is worthwhile to mention again that none of the general approaches to default reasoning with specificity proposed in the literatures up to now captures nonmonotonic inheritance reasoning and at the same time satisfies core properties of default reasoning. We also provide a transformation from acyclic and consistent default theories into general logic programs where answer set semantics coincides with our newly defined semantics. Thus, the new approach points out that logic program can be used to implement reasoning systems which can deal with specific information correctly. It is worth noting that the transformation is modular. Thus, it can be applied in building large scale knowledge base involving reasoning with specificity. But, many works are still needed, for example, the problem of reasoning with specificity in default theories where the background knowledge is not

empty, i.e., $B \neq \emptyset$ and the set of evidences contains disjunctive evidences. These are also the topics of our next papers.

Acknowledgment

The first author is partially supported by EEC "Keep in Touch" activity KIT011-LPKRR. We would like to thank Michael Gelfond and the anonymous referees for their many constructive critisms, which helped us to improve the paper in many ways.

References

[1] Brewka, G.: Adding Priorities and Specificity to Default Logic, Proc. JELIA 94, LNAI 838, Springer Verlag, 247–260.

[2] Brewka, G.: Cumulative default logic: In defense of nonmonotonic inference rules, AI'50, 183–205.

[3] Baader, F. and Hollunder, B.: How to prefer more specific defaults in terminological default logic, IJCAI-93

[4] Bondarenko A., Dung P.M., Kowalski R.A., Tony F.: An abstract, argumentation-theoretic approach to default reasoning. To appear in AI Journal.

[5] Delgrande , J.P., and Schaub, T.H.: A General Approach to Specificity in Default Reasoning, Proceeding of KR'94, p. 147–158.

[6] Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning and logic programming and N-person game. AI Vol. **77** 2 (1995) 321–357

[7] Dung, P.M. and Son, T.C.: Nonmonotonic Inheritance , Argumentation, and Logic Programming, Proceeding of LPNMR, 1995, 316–329.

[8] Etherington, D., and Reiter, R.: On Inheritance Hierarchies with Exceptions in Proc. of AAAI-83, Pg. 104–108, 1983

[9] Gelfond, M., Lifschitz, V.: The stable model semantics for logic programs, Proceeding of the 5th ICLP, MIT Press, (1988), 1070 – 1079

[10] Gelfond, M., Lifschitz, V.: Logic Programs with Classical Negation, Proceeding of the 7th ICLP, MIT Press, (1990), 579 –597

[11] Geffner, H., and Pearl J.: Conditional entailment: bridging two approaches to default reasoning, AI'53, 209–244, 1992

[12] Geert, P. and Veimert, D.: A nonmonotonic reasoning formalism using implicit specificity information, LPNRM Proceeding, 1994, p. 380–396

[13] Horty, J.F.: Some direct Theories of Nonmonotonic Inheritance in Handbook of Logic and Artificial Intelligence and Logic Programming, D.Gabbay and C. Hogger, Oxford Uni., 1991.

[14] Thomason, R.H., and Horty, J.F: Logics for Inheritance Theory, 2nd Workshop on nonmonotonic reasoning

[15] Horty, J.F., Thomason, R.H., and Touretzky, D.S.: A skeptical theory of inheritance in nonmonotonic semantic networks, AI Vol. 42, pp. 311–348, 1987.

[16] Kraus, S., Lehmann, D., and Magidor, M.: Nonmonotonic Reasoning, Preferential Models and Cumulative Logics. AI 44 (1990) 167–207

[17] Moore, R.C.: Semantical Considerations on Nonmonotonic Logics, AI'25, 75–94

[18] McCarthy, J.: Circumscription - a form of nonmonotonic reasoning , AI'13, 27–39

[19] Matkinson, D., Schlechta K.: Floating conclusion and zombie paths: two deep difficulties in the 'directly skeptical' approach to defeasible inheritance nets, AI Vol. 48, pp 99–209

[20] McDermott, D. and Doyle, J.: Nonmonotonic Logic I, AI'13, 41–72

[21] Lin, F.: A study of nonmonotonic reasoning, Ph.D. Dissertation, Stanford University. (1991)

[22] Lifschitz, V.: Circumscription, in Handbook of Logic and Artificial Intelligence and Logic Programming, D.Gabbay and C. Hogger, Oxford University, 1994, pp 297–351.

[23] Lloyd, J.W.: Foundations of Logic Programming, Springer Verlag, 1987.

[24] Pearl, J.: System Z: A natural ordering of defaults with tractable applications to nonmonotonic reasoning: A survey; Proc. KR-89, Toronto, Morgan Kaufman, 505–516

[25] Pollock, J.L.: Defeasible reasoning. Cognitive Science **17** (1987) 481–518

[26] Poole, D.: On the comparison of theories: Preferring the most specific explanation, Proceeding IJCAI-85,pp.144–147

[27] Poole, D: A logical framework for default reasoning, AI 36, 1988

[28] Prakken, H., and Sartor, G.: A semantics for argument-based systems with weak and strong negation and with explicit priorities. Draft, July 1995. An extended abstract can be found in Proceedings of the 5th International Conf. on AI and Laws, ACM Press, 1995, 1-9.

[29] Reiter R.: A Logic for Default Reasoning, in Readings in Nonmonotonic Reasoning, Edited by M. L. Ginsberg, Morgan Kaufmann Publishers, Inc., Los Altos, California (1987) 68–93

[30] Simary, G.R., and Loui, R.P.: A mathematical treatment of defeasible reasoning and its implementation, Elsevier, AI, pp. 125–257.

[31] Simonet, G.: RS theory: a really skeptical theory of inheritance with exceptions, Proceeding of ECAI (1990) 615–626.

[32] Stein, L.A.: Resolving ambiguity in nonmonotonic inheritance hierarchies, AI'55, 259–310.

[33] Touretzky, D. S., Horty, J.F., and Thomason, R.H.: A clash of Intuition: The current state of Non-monotonic Multiple Inheritance Systems, IJCAI, 1987, pp.476–482

[34] Touretzky, D. S., Horty, J.F., and Thomason, R.H.: A skeptic's Menagerie: Conflictors, Preemptors, Reinstaters, and zombies in nonmonotonic Inheritance, IJCAI, 1991, pp.478–483

[35] Touretzky, D. S.: The mathematics of Inheritance Systems, Morgan Kaufmann Pub., Inc., Los Altos, CA, 1986.

[36] Van Gelder A., Ross, K., Schlipf J.S.: Unfounded sets and well-founded semantics for general logic programs. Proceeding of PODS 1988.

[37] Vreeswijk, G.A.W.: Studies in Defeasible Argumentation, Ph.D Thesis, Vrije University, Holland

Appendix 1

Lemma 6.1 *Let $T = (E, D)$ be an acyclic and consistent default theory with $E = hd(d)$, $d \in D$. Then, the argument $A = \{d\}$ is not attacked by specificity.*

Proof Assume the contrary that there is an argument B that attacks A by specificity. Then, by definition there is a specificity relevant MCS $C_{d'}$ such that d, d' are the conflicted defaults in $C_{d'}$. By definition 4.11 there is some $S \subseteq D$ such that $bd(d) \vdash_S bd(d')$. On the other side, from $d \in C_{d'}$ and lemma 4.1 we have that there exists $S' \subseteq D$ such that $bd(d') \vdash_{S'} bd(d)$. Because T is acyclic we have $bd(d) \equiv bd(d')$. Hence T is inconsistent. Contradiction. !!! □

Now, let a be a ground atom. We define the *degree of a*, denoted by *degree*(a), as the maximum of the lengths of the directed paths $a_0 \rightarrow \ldots \rightarrow a_n$ in the atom dependency graph of T such that $a = a_n$ and a_0 is the atom of some literal in E. The degree of a literal l is defined by *degree*(l) (resp. *degree*$(\neg l)$) if l is a positive (resp. negative) literal. The *degree of an argument* is defined as the maximum of the degrees of its conclusions.We then have the following lemma.

Lemma 6.2 *Let T be an acyclic and consistent default theory and A, B are two arguments in T, A attacks B by specificity. Then, there is an argument A' that attacks B by specificity s.t. $deg_T(A') < deg_T(B)$.* □

Proof of Theorem 1 Let W be a preferred extension of AF_T and A be an argument, $A \notin W$. We prove by induction on $deg_T(A)$ that W attack A. The theorem is trivial for $deg_T(A) = 0$ because there is no argument $A \notin W$ with $deg_T(A) = 0$. Assume that we have proved for $deg_T(A) = k$. We now prove it for $deg_T(A) = k + 1$. If W attacks A then the theorem is proved. Now, assume that W does not attack A. Consider $W' = W \cup \{A\}$. Clearly, if there is no argument B that attacks A by specificity then A is defendable in W'. Hence, W is not a preferred extension of AF_T. Contradictory !!! Thus, there exists some argument B that attacks A by specificity. Because of lemma 6.2 we know that there exists some argument A' such that $A' \subseteq B$ which is more specific than A. So, $deg_T(A') < deg_T(A)$ and A' attacks A by specificity. Since W does not attack A, A' does not belong to W. By inductive hypothesis, W attacks A' and therefore W attacks B. This holds for every argument B that attacks A by specificity. Hence, A is defendable in W'. Thus, W is not a preferred extension of AF_T. Contradictory !!! It follows that W attacks A. □

Appendix 2

Lemma 6.3 . *Let $T = (D, E)$ be a acyclic default theory with $T \mid\sim a$. Let $T' = (D, E \cup \{a\})$. Then, $T \mid\sim b$ if $T' \mid\sim b$*

Proof Let AR_T and $AR_{T'}$ be the set of arguments of T and T' respectively. Further, let DA_1 be the set of arguments in T but not in T' and DA_2 be the set of arguments in T' but not in T. It is easy to see that $DA_1 = AR_T \setminus AR_{T'} = \{A \ s.t. \ \neg a \in Con(A)\}$ and $DA_2 = AR_{T'} \setminus AR_T = \{A \ s.t. \ \exists d \in A, \ E \not\vdash_A bd(d)\}$. Now, let S be a preferred extension of T. Since $T \mid\sim a$ there exists an argument A supporting a and also there is no argument supporting $\neg a$ in S. Hence, $S \subseteq AR_{T'}$. We define $S' = S \cup \{A' \in DA_2 \ s.t. \ \exists A, \ A$ supports a, $A \cup A' \in S\}$ and prove that S' is a preferred extension of T'. We first prove that S' is conflict free. Assume the contrary there are P and Q in S' and P attacks Q. There are two cases: 1. $P \in S$ and $Q \in DA_2$. Then, P attacks $Q' = A \cup Q$ in T and $Q' \in S$. Contradictory !!! 2. $P \in DA_2$ and $Q \in DA_2$. Then, $P' = A \cup P$ attacks $Q' = A \cup Q$ in T and $Q' \in S$. Contradictory !!! So, we have that S' is conflict free (i). Secondly, we prove that S' attacks every argument that does not belong to S'. Assume that $P \notin S'$. Then either $P \in AR_T$ or $P \in DA_2$. If $P \in AR_T$ then S attacks P (ii). If $P \in DA_2$ then then either A, P are in conflict (ii), or $A \cup P \notin S$ (iii). So, A attacks P (case ii) or S attacks $A \cup P$ (case iii). Hence, S' attacks P. From (i)-(iii) and from theorem 1 we have that S' is a preferred extension of T'. The lemma is proved. □

Lemma 6.4 *Let $T = (E, D)$ be a default theory and A be a minimal (wrt set inclusion) argument supporting a, i.e., $E \vdash_A a$ and $\forall B \subset A, E \not\vdash_B a$. Then, there is a path from atom$(hd(d))$ to atom(a) in G_T for every $d \in A$.* □

In following, by $T_a = (E, D')$ we denote the default theory which is obtained from a default theory $T = (E, D)$ by removing all the default $d \in D$ such that there is no path from $atom(hd(d))$ to $atom(a)$ in G_T. Obviously, an argument in T_a is an argument in T but not vice versa. Further, given an argument A, we define B as a subargument of A if $B \subseteq A$. It follows immediately the next lemma.

Lemma 6.5 *Let $T = (E, D)$ be a default theory and a be a ground literal in the language $\mathcal{L}$ and A be an argument in T_a. Then, each argument B in T which attacks A contains a subargument B' which belongs to AR_{T_a} and attacks A.*

Proof Assuming B' is a minimal subargument of B that attacks A (no subargument of B' attacks A). If B' is in conflict with A then there exists a literal l such that $l \in Con(A)$ and $\neg l \in Con(B')$. Because $A \in AR_{T_a}$ there is a path from $atom(l)$ to $atom(a)$. Hence, by lemma 6.4 we have that B' is an argument in T_a. If B' attacks A by specificity then lemma 4.1 indicates that there is a specificity relevant MCS C_d with d' being the other conflicted default and $d' \in A$ and $C_d \setminus \{d', d\} \subseteq B'$ and $bd(d) \subseteq Con(B')$. From the minimality of B' and lemma 4.1 it is easy to see that for $d \in B'$ there is some path from $atom(d)$ to $atom(d')$ from which there is a path to $atom(a)$. Thus, B' is an argument in T_a. The lemma is proved. □

The next lemma follows from Lemma 6.5.

Lemma 6.6 *Let T be a default theory and a be a literal in $\mathcal{L}$ and T_a is defined as above. Then, we have:*

1. *If W is a stable extension of T then $W_a = W \cap AR_{T_a}$ is a stable extension of T_a.* □
2. *If W_a is a stable extension of T_a then there exists a stable extension W of T such that $W_a \subseteq W$.* □

Lemma 6.7 *Let $T = (E, D)$ be a default theory such that $T \mid\sim a$. Then, each preferred extension of $T' = (E \cup \{a\}, D)$ contains a nonempty argument $A \in AR_T$ supporting a (wrt T).*

Proof Consider the default theory T'_a. It is easy to see that $AR_{T'_a} \subseteq AR_{T_a}$ and $AR_{T_a} \setminus AR_{T'_a} = \{A \ s.t. \ A \in AR_{T_a}, A \text{ supports } \neg a\}$. Further, by infiniteness of T there exists only finitely many arguments $A_1, \ldots, A_n$ in T_a which support a. Assuming the contrary, there exists a preferred extension S of T' which does not containing an argument A, $A \neq \emptyset$, which supports a. Let S_a be the set of arguments in S which belong to $AR_{T'_a}$. Lemma 6.6 implies that S_a is a preferred extension of T'_a. Since $S_a \subseteq AR_{T'_a}$ and $AR_{T'_a} \subseteq AR_{T_a}$ we have $S_a \subseteq AR_{T_a}$. We now prove that S_a is an admissible set of arguments in T_a. Clearly, S_a is conflict free. Thus, we only need to prove that if $B \in AR_{T_a}$, B attacks S_a then S_a attacks B. There are two cases: 1. B does not support $\neg a$. Then, $B \in AR_{T'_a}$ and hence S_a attacks B (i). 2. B supports $\neg a$. Then, B does not attack S_a by conflict at a because there is no argument supporting a in S_a. From lemma 6.5 we can conclude that there is a subargument B' of B which attacks S_a and $B' \in AR_{T_a}$. Thus, S_a attacks B' and hence S_a attacks B (ii). From (i)-(ii) we can conclude that S_a is admissible in T_a. Thus, from the result in [6] there is a preferred extension W of T_a such that $S_a \subseteq W$. Since S_a attacks every A_i which supports a so W cannot contain an argument supporting a. Contradictory because $T_a \mid\sim a$ (lemma 6.6). Thus, our assumption is incorrect. Hence, the lemma is proved. □

The next lemma follows directly from lemma 6.7.

Lemma 6.8 *Let $T = (E, D)$ be a default theory such that $T \mid\sim a$. Further let $T' = (E \cup \{a\}, D)$ and W' be a preferred extension of T'. Then $W = W' \cap AR_T$ is a preferred extension of T* □

Appendix 3

Let E be a credulous extension of Γ. Then, the set of ground paths in E, denoted by E_G, is called a *ground credulous extension* of Γ.

For a ground path $\alpha = \pi(a, x_1, ..., x_n)$ in Γ, A_α denotes the set consisting of ground defaults $x_i(a)/x_{i+1}(a)$, $1 \leq i \leq n-1$.

For a set of ground paths Φ, let $A_\Phi = \bigcup_{\alpha \in \Phi} A_\alpha$.

Given an argument B in T_Γ we define $\Phi_B = \{\alpha \mid \alpha \text{ is a ground path in } \Gamma, \text{ s.t. } A_\alpha \subseteq B\}$ and for a set W of arguments we write $\Phi_W = \bigcup_{B \in W} \Phi_B$.

It is easy to see that if Γ is acyclic and consistent then T_Γ is also acyclic and consistent. And, if α be a ground path in Γ then A_α is an argument in T_Γ. Also, it is clear that if a ground literal l is supported by an argument A then there exists a ground path $\alpha \in \Phi_A$ such that $c(\alpha) = l$. In the next sequel a path $\alpha = \pi(x, \sigma, u)$ is called a *prefix* of a path $\beta = \pi(x, \sigma, u, \tau, v)$ in Γ.

A ground path $\alpha = \pi(a, .., x)$ is said to *support the conclusion* $x(a)$ (resp. $\neg x(a)$). $x(a)$ (resp. $\neg x(a)$) is called the conclusion of α, denoted by $c(\alpha)$.

For a set Φ of ground paths, $c(\Phi) = \{c(\alpha) \mid \alpha \in \Phi\}$.

Lemma 6.9 *Let Γ be an acyclic and consistent inheritance network and Φ be a credulous extension of Γ. Then, $W = \{A \mid A \text{ is an argument}, A \subseteq A_\Phi\}$ is a preferred extension of T_Γ.*

Proof It is easy to see that A_Φ is an attack-free argument. We first prove that A_Φ attacks every argument B that attacks A_Φ. Consider two cases: 1. B attacks A_Φ by conflict, then A_Φ attacks B by definition. 2. B attacks A_Φ by specificity. It means there is a path $\alpha \in \Phi$ s.t. α is preempted in Φ_B.

Without the lose of generality we assume that $\alpha = \pi(x, \ldots, u, y)$. α is preempted in Φ_B means there exists $\beta = \pi(x, \ldots, v, \ldots, u) \in \Phi_B$ and $u \not\rightarrow y \in \Gamma$. Because $\alpha \in \Phi$ we have $\beta \notin \Phi$. Hence, there exists a prefix δ of β which is conflicted or preempted in Φ. It follows then A_Φ attacks B by conflict or by specificity. Thus, A_Φ attacks every argument B that attacks A_Φ. Hence, for all $A \in W$, A is defendable in W (i). Now, let B be an argument and $B \notin W$. Trivially, there exists some path $\alpha \notin \Phi$ s.t. $A_\alpha \subseteq B$, $A_\alpha \setminus A_\Phi \neq \emptyset$ and therefore A_Φ attacks A_α. Thus, A_Φ attacks B (ii). From (i)-(ii) we can conclude that W is a preferred extension of T_Γ. □

Lemma 6.10 *Let W be a preferred extension of T_Γ. Then, Φ_W is a ground credulous extension of Γ.*

Proof Let σ be a ground path in Γ. We prove that (i) if $\sigma \in \Phi_W$ then σ is defeasibly inheritable in Φ_W and (ii) if $\sigma \notin \Phi_W$ then σ is not defeasibly inheritable in Φ_W. We first prove (i). It follows from $A_\sigma \subseteq W$ that $A_\delta \subseteq W$ if δ is a prefix of σ. Hence, all prefixes of σ belong to Φ_W. So, σ is constructible in Φ_W. Furthermore, W is a preferred extension implies that σ is not conflicted or preempted in Φ_W. Thus, $\Phi_W \mid\sim \sigma$ for $\sigma \in \Phi_W$ (1). Now, consider $\sigma \notin \Phi_W$. Without the lose of generality we assume that all prefixes of σ are in Φ_W. It follows that A_σ is attacked by W. If A_σ is attacked by conflict then σ is conflicted in Φ_W. If A_σ is attacked by specificity then it is preempted in Φ_W. Thus, $\Phi_W \not\mid\sim \sigma$ for $\sigma \notin \Phi_W$ (2). From (1) and (2) we can conclude that Φ_W is a ground credulous extension of Γ. The lemma is proved. □

Theorem 5 follows from lemma 6.9 and lemma 6.10 .

Appendix 4

Let I_T be the set of constants in T and AB_T denotes the set of all ground positive literals of abnormal-predicates in P_T. Further, for a set W of arguments in T we write $Con(W) = \bigcup_{A \in W} Con(A)$. For a preferred extension W of T let $Ab_W = \{ab_j(a) |\ a \in I_T, d_j \in D, bd(d_j) \subseteq Con(W),\ hd(d_j) \notin Con(W)\}$ We prove that

Lemma 6.11 *Let W be a preferred extension of T. Then, $S = Con(W) \cup Ab_W$ is an answer set of P_T.*

Proof S is an answer set of P_T [10] if S is the answer set of $P_T(S)$ which is obtained by (i) deleting all clauses containing one literal *not* $ab_j(a)$ if $ab_j(a) \in S$ and (ii) deleting all the abnormal-literals in the remaining clauses. In following let $\alpha(P_T(S))$ be the answer set of $P_T(S)$. By definition of $Con(W)$ and Ab_E, we prove the lemma by two cases:

- Case: $q(a) \in S$
  iff there is a sequence σ of $d_1, \ldots, d_n$ in W s.t. $hd(d_n) = q(a)$ and $E \vdash_\sigma bd(d_i)$ for $1 \leq i \leq n$
  iff $ab_j(a) \in (AB_T \setminus Ab_W)$ for all d_j
  iff $ab_j(a) \notin S$ for all defaults j belongs to σ
  iff $P_T(S)$ consists of $hd_j(a) \leftarrow bd_j(a)$ for all d_j belongs to σ
  iff $q(a) \in \alpha(P_T(S))$.

- Case: $ab_j(a) \in S$
  iff $ab_j(a) \in Ab_W$
  iff $\exists d_j \in D$, $bd_j(a) \subseteq Con(W)$ but $hd_j(a) \notin W$
  iff there is an argument $A \in W$ but $B = A \cup \{d_j\} \notin W$
  iff B is attacked by W
  iff $ab_j(a) \leftarrow \neg hd_j(a)$ is a clause of $P_T(S)$ and $\neg hd_j(a) \in S$
  or $ab_j(a) \leftarrow \bigcup_{k \in C_i} hd_k(a)$ is a clause of $P_T(S)$ and $\forall k \in C_i :\ hd_k(a) \in S$
  iff $ab_j(a) \in \alpha(P_T(S))$.

For a set of ground literal M of the program P_T we define $A_M = \{A \mid A$ is an argument in T s.t. $Con(A) \subseteq M\}$, $P_M = \{A \mid A \in A_M,\ A$ is attacked by A_M by specificity$\}$. We show that

Lemma 6.12 *Let M be an answer set of P_T. Then, $A_M \setminus P_M$ is a preferred extension of T.*

Proof Let $W = A_M \setminus P_M$. It is easy to see that there exists no argument $A \in A_M$ such that A attacks A_M by conflict because M is consistent. Hence, W is attack-free. Consider an argument A in T, if $A \in W$ then we can prove that W attacks B if B attacks A. Otherwise, $A \notin W$. It means that there exists some default $d_i \in A \setminus W$. So, $ab_i \in M$. There are two cases: (i) $\neg h_i$ is supported in M or (ii) there is a specific relevant MCS C such that d_i is one of the conflicted default but not the most specific default. It is easy to see that in (i) W attacks A by conflict and in (ii) W attacks A by specificity. Thus, W attacks A if $A \notin W$. Hence, the lemma is proved. □

From lemma 6.11, 6.12 we have Theorem 6.

Proof of Theorem 8 According to [10] we need to prove that there is no consistent stable model S of P'_T and a be an arbitrary ground atom in the language of P_T such that $\{a, a'\} \subset S$. Assume the contrary. Since there is no negative abnormal literal in P_T we can conclude that a is a ordinary literal in the language $\mathcal{L}$ of T. Further, because T is consistent $\{a, a'\} \subset S$ if and only if there exists a constant k and two ground instances of P'_T such that $hd_i(k) \leftarrow bd_i(k), ab_i(k)$ and $hd_j(k) \leftarrow bd_j(k), ab_i(k)$ such that $a = hd_i(k)$, $a' = hd_j(k)$, and $bd_i(k) \subset S$, $bd_j(k) \subset S$, $\{ab_i(k), ab_j(k)\} \cap S = \emptyset$ (*). But, by definition of P_T we have that $ab_i(k) \leftarrow hd_j(k)$ and $ab_j(k) \leftarrow hd_i(k)$ are ground instances of P'_T. Hence, S must include $\{ab_i(k), ab_j(k)\}$ if S is a stable model. Contradictory to (*). !!! The theorem is proved. □

Default Reasoning System DeReS

Paweł Cholewiński*
Computer Science Dept.
University of Kentucky
Lexington, KY 40506-0046

Victor W. Marek
Computer Science Dept.
University of Kentucky
Lexington, KY 40506-0046

Mirosław Truszczyński
Computer Science Dept.
University of Kentucky
Lexington, KY 40506-0046

Abstract

In this paper, we describe an automated reasoning system, called DeReS. DeReS implements default logic of Reiter by supporting several basic reasoning tasks such as testing whether extensions exist, finding one or all extensions (if at least one exists) and querying if a formula belongs to one or all extensions. If an input theory is a logic program, DeReS computes stable models of this program and supports queries on membership of an atom in some or all stable models. The paper contains an account of our preliminary experiments with DeReS and a discussion of the results. We show that a choice of a propositional prover is critical for the efficiency of DeReS. We also present a general technique that eliminates the need for some global consistency checks and results in substantial speedups. We experimentally demonstrate the potential of the concept of relaxed stratification for making automated reasoning systems practical.

1 INTRODUCTION

The area of nonmonotonic logics was originated about twenty years ago [Rei78, Rei80, MD80, McC80] in an effort to build efficient knowledge representation formalisms. Since then solid theoretical foundations of nonmonotonic logics have been established. The efforts of the past two decades culminated in several research monographs [Eth88, Bes89, Bre91, MT93] describing major nonmonotonic systems: default logic, logic programming with negation as failure, autoepistemic logics and circumscription.

*Current address: Sagent Corporation, 11201 SE 8th St. J140, Bellevue, WA 98004-6420

It was expected that nonmonotonic logics would be able to model formal aspects of commonsense reasoning and that their computational properties would be better than those of classical logics. Computational complexity results obtained in recent years were, however, discouraging. Decision problems associated with nonmonotonic reasoning, even when restricted to the propositional case, are very complex. For example, in the case of logic programming with stable model semantics they are NP-complete or co-NP-complete. In the case of default logic, they are Σ_2^P-complete or Π_2^P-complete.

However, the complexity results do not necessarily disqualify nonmonotonic logics as a computational knowledge representation mechanism. Decision problems in classical logic are also highly computationally complex — NP- and co-NP-complete in the propositional case. These tasks become even more complex in the case of quantified Boolean formulas [MS72]. In the same time, recent experimental results on satisfiability indicate that propositional logic can serve as a computational tool, and is capable of handling large collections of variables and clauses [SLM92, SKC93]. Complexity results alone are also insufficient to determine whether classical logic or nonmonotonic logics are better suited as the basis for an automated reasoning system. In particular, the results of [CDS94, GKPS95] show that higher computational complexity of nonmonotonic logics may be offset by more concise encodings than those possible with propositional logic.

Hence, despite the tremendous progress in our understanding of the basic principles of nonmonotonic logics, a major question of their viability as a computational tool remains unresolved. Systematic implementation and experimentation effort is necessary to provide us with better insights into the computational properties of nonmonotonic logics. Despite evident importance of experimental studies of nonmonotonic logics, there has been little work re-

ported in the literature. While several algorithms were published and some implementations described [MW88, BNN+93, BNN+94, BEP94, NS95] the results are far from conclusive. This state of affairs can be attributed to the lack of systematic experimentation with implemented systems. A possible reason for this was the absence of commonly accepted benchmarking systems. In particular, a convincing and useful for experimentation model of a random logic program has yet to be proposed.

To address the issue of benchmarks we developed a system, called TheoryBase, generating propositional logic programs and default theories [CMMT95]. This system allows the users to systematically generate interesting families of logic programs and default theories for experimentation. A key feature of TheoryBase is that it provides an identifier (label) for every theory it generates. This can greatly facilitate experimental comparisons between implemented systems. TheoryBase has already been used by several research groups around the world. Most notably, Niemelä and Simons [Nie95, NS95] used the ideas behind TheoryBase to study their implementation of a system for computing stable models of logic programs.

In this paper we report on the project to design and implement a program, Default Reasoning System DeReS, that supports basic automated reasoning tasks for default logic [Rei80] and for logic programming with stable model semantics [GL88]. We describe basic features of DeReS. We also present experimental results obtained by running DeReS on default theories encoding graph problems generated by TheoryBase[1].

Our current version of DeReS is built around the notion of *relaxed stratification* [Cho95c, Cho95a]. Relaxed stratification allows us to use a *divide-and-conquer* approach when computing extensions. An original default theory is partitioned into several smaller subtheories, called *strata*. The extensions of the original theory are then reconstructed from the extensions of its strata. The notion of relaxed stratification considered here is a generalization of the original concept as introduced in [ABW88]. In particular, a theory stratified in our sense may possess no extension (stable model) or, if it does, not necessarily a unique one. In the paper, we show that applying relaxed stratification leads to dramatic speedups, especially when the strata are small.

[1]To date, both systems: TheoryBase and DeReS, have been installed and tested in Bern, Ithaca, Karlsruhe, Lexington, Riverside and San Diego. They are currently available by anonymous ftp from al.cs.engr.uky.edu. The file names are: `TheoryBase.tar.gz` and `DeReS.tar.gz` in the directory `/cs/software/logic`.

In the paper we also study the effects of different propositional theorem provers on the efficiency of DeReS. We observe that full theorem provers, which check for global consistency when deciding whether a theory proves a formula, result in performing prohibitive amount of redundant computation. A weaker notion of a *local prover*, sound but not complete, can also be used to correctly implement DeReS and results in substantial improvements in time performance.

Our results show that there are classes of theories that DeReS can handle very efficiently. However, if relaxed stratification does not yield a partition of an input theory into small strata, the efficiency of DeReS may be poor. In this context, it is interesting to relate our work to the work of Niemelä and Simons [NS95]. The two implementations are difficult to compare as they attack different aspects of the same problem. While our research focused on techniques to exploit relaxed stratification and limit the number of global consistency checks, Niemelä and Simons studied techniques to deal with individual strata. It seems that the next generation implementations of nonmonotonic systems, in order to be effective in a large range of different applications, must combine the techniques developed in both projects.

The paper is organized as follows. In the next section, we describe DeReS. We briefly describe basic properties of relaxed stratification, introduce local provers and present the benchmark problem generator, TheoryBase. Then, in Section 3 we show how to use these tools in our implementation of DeReS. We also present and discuss the results of our experimentation. The last section contains conclusions.

2 OVERVIEW OF DeReS

DeReS is a comprehensive software package for nonmonotonic reasoning. The focus of DeReS is on automated reasoning with default logic and logic programming (although some other nonmonotonic formalisms are also supported). The programs are written in C and run under UNIX operating system. The package was developed on a Sun SPARCstation 20 with the SunOS 5.4 operating system. Nevertheless, no special features of this environment are used and DeReS can be installed on any machine running SYSV or BSD version of UNIX. The main components of DeReS are:

user interface — the module facilitating interactive work with stored default theories,

scanner — the module accepting default theory, checking for errors, and building internal data

structures,

default engine — a library of routines for reasoning with a given default theory,

provers — the module providing several propositional theorem provers that can be called by the default engine module,

graphical interface — an optional graphical interface for displaying the progress of computation and the results.

DeReS computes extensions for finite propositional default theories. There are no syntactic restrictions on formulas which can occur in the input. DeReS can compute the list of all extensions for a given default theory or check whether a given default theory has an extension. It also outputs basic time characteristics for each solved query – the total time spent processing the query and the number of calls to propositional provability routine. The user communicates with DeReS via its shell, which accepts user commands and starts desired tasks.

We will now focus on the main aspects of DeReS. We will also provide a short overview of TheoryBase. Two main questions that we study are:

1. What is the effect of propositional provers on the efficiency of DeReS?
2. What is the role of relaxed stratification in efficient implementations of nonmonotonic logics?

These questions are further expanded in Sections 2.1 and 2.2. We discuss TheoryBase in Section 2.3. Section 3 contains the description of the experiments and our findings.

2.1 Propositional provers

The prover module of DeReS system is used as a propositional provability oracle by all reasoning routines. Since the computational complexity of basic decision problems in nonmonotonic reasoning is very high, special care is needed to design efficient provers, or to design techniques to use provers more efficiently. DeReS is equipped with the following propositional provability procedures:

Full prover — sound and complete propositional tableaux theorem prover,

Local prover — local provability propositional tableaux theorem prover (sound but not complete),

df-lookup — table lookup method for disjunction-free theories.

Full prover, implemented using the propositional tableaux method, allows DeReS to work with arbitrary propositional theories. Local prover is also based on the tableaux method. It differs from the full prover in that it does not, in general, perform global consistency checks. Finally, a *df-lookup prover* is a table lookup method applicable to *disjunction-free* theories. A default theory (D, W) is disjunction-free if all formulas in W, all prerequisites, justifications and conclusions of defaults in D are conjunctions of literals. For disjunction-free theories provability problems can be decided by determining membership of a literal a in a set of literals T (assuming that T does not contain contradictory literals — in such a case, the df-lookup returns **true**). The df-lookup can be implemented to run in time $O(1)$, which results in dramatic performance improvement. The class of disjunction-free default theories is simple yet quite powerful. In particular, it subsumes the class of extended logic programs.

We will now describe the idea behind a local prover. Although this approach does not improve the worst case complexity of the reasoning algorithms it usually yields significant speedups. Let $\mathcal{L}$ be any propositional language. Consider a propositional theory T contained in $\mathcal{L}$. Let φ be a formula in $\mathcal{L}$ and let $Var(\varphi)$ denote the set of propositional variables in the formula φ.

A theory T proves a formula φ (in symbols, $T \models \varphi$) in standard propositional logic if either the information contained in T and concerning the propositional variables occurring in φ allows us to derive φ, or if T is inconsistent. This observation motivates the following definitions.

Definition 2.1 *Let T be a set of propositional formulas. The* incidence graph G_T *is a simple undirected graph $G_T = (T, E)$ such that for any $\varphi, \psi \in T$*

$$\{\varphi, \psi\} \in E \quad \textit{if and only if} \quad Var(\varphi) \cap Var(\psi) \neq \emptyset$$

Let $\varphi \in T$. By T_φ we denote the set of vertices of the connected component of G_T which contains φ.

Intuitively, local provability of φ by a theory T means that φ is entailed by the part of T consisting of formulas containing variables relevant to φ, and not because T contains inconsistent data[2]. A careful formalization of this idea yields the following definition.

[2]Eliminating global consistency checks by local provers is analogous to omitting occur check in logic programming.

Definition 2.2 *A theory T* locally proves *a formula φ (denoted $T \models_{loc} \varphi$) if $(T \cup \{\varphi\})_\varphi \setminus \{\varphi\} \models \varphi$.*

Let us consider now standard propositional provability routines, such as tableaux or resolution based algorithms. Suppose we want to use them to answer the query $T \models_{loc} \varphi$. The modifications required in these algorithms to implement local provability are very simple. All one has to do is to block expanding the tableaux tree or the resolution list by formulas which have no common variables with the variables mentioned in the structure (the tree or the list) so far. This way the prover remains restricted to the component of $G_{T\cup\{\varphi\}}$ containing the formula φ.

Clearly, local provers are faster than full provers as, in general, they do not consider all formulas in the input theory. The exact amount of savings depends on the structure of the underlying theory T and the input formula φ or, more precisely, on the structure of the incidence graph $G_{T\cup\{\varphi\}}$. We gather some immediate consequences of Definition 2.2 in the following theorem.

Theorem 2.1 *For arbitrary theory T and formula φ the following holds:*

1. *If $T \models_{loc} \varphi$ then $T \models \varphi$.*
2. *If T is consistent then*

   $$T \models \varphi \text{ if and only if } T \models_{loc} \varphi.$$

3. *If $G_{T\cup\{\varphi\}}$ is connected then*

   $$T \models \varphi \text{ if and only if } T \models_{loc} \varphi.$$

4. *$T \models \varphi$ if and only if either T is inconsistent or $T \models_{loc} \varphi$.* □

Theorem 2.1 yields a technique to compute extensions using a local prover directly. Let (D, W) be an input default theory. Assume that the task is to find an extension (or generate all of them). To simplify the discussion, we will assume that each default in D has at least one justification. Under this assumption, (D, W) has either a unique inconsistent extension (if W is inconsistent), or all extensions of (D, W) are consistent (if W is consistent). Hence, a single call to a full propositional prover is sufficient to decide which of the two possibilities holds. If it is the first one, we stop. So, from now on we will assume that W is consistent.

It is well known that extensions of a default theory (D, W) are determined by the sets of formulas of the form $W \cup c(U)$, where $U \subseteq D$ and $c(U)$ consists of the consequents of defaults in U ([MT93]). In order to verify whether $W \cup c(U)$ generates an extension, DeReS has to decide several provability problems of the form $W \cup c(U) \vdash \varphi$. If $W \cup c(U)$ is consistent, the local prover can be used instead of the full prover.

To search through all such sets, DeReS considers systematically all subsets U of D, starting with $U = \emptyset$. Then $W \cup c(U) = W$ which, by our assumption, is consistent. Hence, the local prover can be used to decide if $W \cup c(U)$ generates an extension. Each next set of defaults, say U', is obtained from the current set U by removing or adding one default, say d. In the first case, the resulting set of formulas $W \cup c(U')$ is consistent and the local prover can be used to decide whether it generates an extension. In the second case, $W \cup c(U')$ is consistent if and only if $W \cup c(U) \not\vdash \neg c(d)$. Since $W \cup c(U)$ is consistent, this condition can be tested by a call to the local prover. If $W \cup c(U')$ is inconsistent, DeReS removes U', as well as all its supersets, from the search space and backtracks to $W \cup c(U)$. Otherwise, $W \cup c(U')$ is consistent and, again, the local prover can be used to test whether $W \cup c(U')$ is an extension.

2.2 Relaxed stratification

To improve efficiency of reasoning with default logic one can check if a given theory belongs to a subclass for which reasoning can be performed faster. A common technique is to stratify a theory. Stratification consists of partitioning a given theory into a sequence of smaller theories for which extensions can be computed faster. This approach was studied in the cases of logic programming [CH82, ABW88, VG88, Prz88] and autoepistemic logic [Gel87, MT91]. Some results of applying this approach to default logic were obtained by Etherington [Eth88] and Kautz and Selman [KS89]. DeReS uses a relaxed variant of stratification applicable to arbitrary default theories. That is, no syntactic restrictions on formulas appearing in defaults are imposed. Instead more restrictive conditions on dependencies between defaults are required. Strata are computed as strong components of a directed graph capturing dependencies between defaults in a default theory. See [Cho95c] and [Cho95a] for details on relaxed stratification for default theories.

Given an input default theory (D, W) and its relaxed stratification $D_1, \ldots, D_k$, DeReS builds extensions gradually, dealing with consecutive strata "one at a time". DeReS assumes a fixed linear order (topological sort) of strata. The nodes of the search tree are pairs (D_i, E) where D_i is the current stratum and E is an extension for the default theory $(D_1 \cup \ldots \cup D_{i-1}, W)$.

The children of this node are of the form (D_{i+1}, E') where D_{i+1} is the next stratum and E' is an extension for the default theory (D_i, E). It is known [Cho95c] that all such extensions are precisely the extensions for $(D_1 \cup \ldots \cup D_i, W)$. DeReS uses an additional (local) search tree to find extensions for each default theory (D_i, E).

2.3 TheoryBase

To test and experiment with an automated reasoning system, one requires a diversified collection of benchmark theories. In our case, to test DeReS, we needed default theories and logic programs. They should be easily generated, realistic and meaningful. They should be easy to reproduce and disseminate to facilitate experimental comparisons of different implementations of nonmonotonic reasoning systems. These requirements prompted another project – TheoryBase [CMMT95]. TheoryBase automatically generates families of default theories and logic programs. It is based on the observation that graph problems such as existence of colorings, kernels and hamiltonian cycles can be encoded as default theories and logic programs, with extensions (stable models) corresponding to combinatorial objects in question. Hence, the TheoryBase first generates a graph and then produces an encoding — a default theory or a logic program — of a combinatorial problem for this particular graph. Several such encodings were described in [CMMT95]. Most important of them are:

1. existence of a kernel in a directed graph,
2. existence of a vertex k-coloring in a graph,
3. existence of a hamiltonian cycle in a directed graph.

In order to generate graphs, TheoryBase uses the Stanford GraphBase, a system developed by Knuth [Knu93]. This software system can generate a large variety of graphs, each with a unique identifier. The system of Stanford GraphBase identifiers is extended in TheoryBase to default theories and logic programs. Each default theory and logic program generated by TheoryBase receives an identifier which allows for an easy reconstruction of the corresponding theory or program.

TheoryBase can easily be extended by new encodings of combinatorial problems as default theories, logic programs, disjunctive logic programs, as well as propositional theories. Our intention is for TheoryBase to evolve into a commonly accepted benchmarking system to support research in automated reasoning.

3 RESULTS

In this section we discuss some of the experiments that were run on DeReS. The test theories were generated by TheoryBase and their identifiers are provided here. Two main questions that we studied were:

1. the impact of a propositional prover on the efficiency of DeReS, and
2. the impact of relaxed stratification on the efficiency of DeReS.

In our tests we focused on two tasks: to decide the existence of extensions, and to generate all extensions of an input default theory.

In the presentation of the results we use the following notation: $time_f$ denotes the CPU time for queries processed with full propositional tableaux, $time_l$ denotes the CPU time for queries processed with local propositional tableaux, $time_a$ denotes CPU time for queries processed with df-lookup prover. Further, NCPP stands for the number of calls to propositional provability routine, CAND stands for the number of candidate theories tested for an extension. We denote the total number of extensions for the input theory by EXT. Clearly, NCPP, CAND and EXT do not depend on the choice of prover. By D we denote the set of defaults of default theories used in experimentation. We set $N = |D|$. Finally, V and E denote the vertex set and the edge set of the graphs underlying default theories used in experimentation.

All times are given in seconds. A bar (–) symbol in tables indicates that the program was running for more than 2 hours without reaching the final answer and then was interrupted. To capture the reasoning time we measure the CPU time from the point when an input default theory is already represented and stored, together with its relaxed stratification, as a DeReS data structure to the point when the answer is returned.

3.1 Computing graph kernels with DeReS

We start our presentation of the experiments with DeReS with the problem of computing kernels in graphs. We considered the family of graphs $G_{n,m} = board(n, m, 0, 0, 5, 3, 1)$ from Stanford GraphBase. The graph $G_{n,m}$ is the graph of chess-knight moves on a wrapped $n \times m$ chessboard. Moreover, the edges are directed according to the lexicographical order on the pairs of coordinates for the chess-board squares.

To compute kernels in the graphs $G_{n,m} = board(n,m,0,0,5,3,1)$ the encodings with identifiers $kernel.board\_n,m,0,0,5,3,1\_$ were generated by TheoryBase.[3] In particular, we studied the graphs $G_{8,m}$. The size of these graphs grows proportionally to m. The same holds for the size of the TheoryBase encodings of the kernel existence problem for these graphs. We propose this sequence of default theories as a benchmark problem for non-monotonic reasoning.

In Table 1 we show times which DeReS needed to find one kernel in graphs $G_{8,m} = board(8,m,0,0,5,3,1)$, for $m = 2,4,6,8,10$. In Table 2 we show the times required to compute all the kernels. In both cases time grows exponentially with the size of the underlying default theory. In the same time, this example illustrates that DeReS is capable to deal with default theories containing hundreds of defaults. It is due to the effective use of relaxed stratification in the encoding Δ^3_{ker}, which results in the partition of the theory into relatively small clusters. Consequently, the provability routine performs very efficiently and the answers are found within seconds.

Let us look more closely at times corresponding to different provability routines. In all cases, the total time required to decide all relevant provability problems using the df-lookup prover is proportional to the total number of calls to the prover. Hence, the time per call is constant. The total time grows exponentially and is of the order $\Theta(3^{N/56})$ (recall that N is the number of defaults in D). That is, it grows at a much smaller rate than the theoretical bound $O(N^2\ 2^N)$ [MT93]. Similarly, the times $time_a(m)$ for finding one extension are of the order $\Theta(2^{N/56})$. These speedups are due to the fact that we were able to stratify input default theories into chains of clusters of defaults and construct extensions by considering several subproblems of small sizes instead of one large set set of defaults.

When tableau provers are used, the times are larger because more time is needed for each call to propositional provability. In the case of local prover, the time required for a single call grows proportionally with m which implies the total time of the order $\Theta(N\ 3^{N/56})$.

Full propositional prover scans the input theory $O(m)$ times per each query. Consequently, the time needed to solve each such query is $O(m^2)$ (quadratic in the size of theory), and the total time of finding all extensions is of the order $\Theta(N^2\ 3^{N/56})$.

These results show that efficiency of provers have substantial effect on the performance of DeReS. One general lesson is that local provers should be used instead of full provers, especially in view of the fact that local provers do not impose any restrictions on input theories. Although our results were obtained for full and local tableaux provers, we believe the same speedups would be obtained if any other full prover is replaced by its local version. That is, consistency checks performed each time when the prover is called can and should be avoided.

Secondly, even though the theories that arise in this example are very simple, both local and full tableaux provers are outperformed by the df-lookup prover. This indicates that when using DeReS as a knowledge representation tool, it is useful to encode domain knowledge as a disjunction-free theory, as this allows DeReS to refer to the df-lookup prover when reasoning.

To discuss in more detail the effects of relaxed stratification, we will consider the same class of graphs but the kernel existence problem will be encoded differently. This time, we will use the encoding Δ^1_{ker}, also described in [CMMT95]. The corresponding theories generated by TheoryBase have identifiers $kernel.b.board\_n,m,0,0,5,3,1\_$ Table 3 contains the experimental results. It is clear that DeReS performs much worse on these theories. The lack of good stratification for the theories $kernel.b.board\_n,m,0,0,5,3,1\_$ is to blame. Hence, encoding domain knowledge in DeReS, as in any other knowledge representation language, requires some skill. In the case of DeReS, the same problem can be encoded efficiently (Δ^3_{ker}) and inefficiently (Δ^1_{ker}). A general lesson is that while encoding domain knowledge in DeReS, encodings that admit fine stratification should be used.

3.2 Coloring ladder graphs

In [NS95], the problem to find a 3-coloring of a ladder graph was proposed as a benchmark for testing non-monotonic reasoning systems. The ladder graph, L_n, is defined as follows. Its vertices are points (x,y) in the real plane such that, $x = 0,1,\ldots,n$ and $y = 0,1$. Two vertices are joined by an edge if the euclidean distance between them equals 1. Graph L_n can be generated using The Stanford GraphBase and has label $board(n,2,0,0,1,0,0)$. We use default theory $chrom_3(G)$ from [CMMT95] to encode 3-colorings for graph G. These encodings were generated using TheoryBase system and have TheoryBase labels $color3.board\_n,2,0,0,1,0,0\_$

DeReS times obtained for the query *find one extension*

[3] Since the theories are saved as UNIX files, the system of TheoryBase identifiers uses underscores instead of parentheses.

kernel.board_8,m,0,0,5,3,1_ one solution								
m	$\|V\|$	$\|E\|$	N	NCPP	CAND	$time_f$	$time_l$	$time_a$
2	16	64	112	1939	851	2.34	0.14	0.01
4	32	128	224	14804	6845	69.95	1.45	0.07
6	48	192	336	121249	56298	1532.87	16.49	0.57
8	64	256	448	308910	143677	-	53.40	1.45
10	80	320	560	1982796	921464	-	421.74	9.14

Table 1: Searching for a kernel in $board(8, m, 0, 0, 5, 3, 1)$.

kernel.board_8,m,0,0,5,3,1_ all solutions									
m	$\|V\|$	$\|E\|$	N	NCPP	CAND	$time_f$	$time_l$	$time_a$	EXT
2	16	64	112	3337	1473	4.03	0.24	0.02	2
4	32	128	224	65704	30016	365.59	6.70	0.32	6
6	48	192	336	421082	192175	5239.54	57.34	1.90	5
8	64	256	448	4130579	1888829	-	720.22	18.97	134
10	80	320	560	31630658	14466688	-	6819.35	141.74	267

Table 2: Computing all kernels in $board(8, m, 0, 0, 5, 3, 1)$.

are shown in Table 4. In this case, the number of calls to propositional provability routine and the number of tested candidates grow linearly with respect to the size (the number of vertices) of the graph. All three provers which we use behave in the same way as in the previous example. That is, df-lookup prover runs in constant time per call, local tableau prover in linear time and full tableau prover in quadratic time. Consequently, the time for finding an extension (coloring of the ladder graph L_n) is $\Theta(n)$ for df-lookup prover, $\Theta(n^2)$ for local tableau and $\Theta(n^3)$ for full tableau method.

Hence, when efficient provability method is used, DeReS can find a 3-coloring for a ladder graph within a second even when the corresponding encoding contains almost 15 thousand defaults. The success of DeReS is again due to the structure of the encoding used — it allows for stratification into small strata. The method for computing stable models of logic programs, developed in [NS95], also did very well for this particular class of problems. Hence, both our results and the results in [NS95] indicate that the 3-coloring problem for ladder graphs is too easy to serve as a useful benchmark problem for nonmonotonic reasoning systems. This is due to the large number of 3-colorings that the ladder graphs admit. Hence, any search method is likely to find a 3-coloring quickly. In fact, there are 3^{n-1} non-isomorphic 3-colorings for L_n. This yields 2×3^n extensions for default theory *color3.board_*n,2,0,0,1,0,0_. This is also the reason why we did not provide time for the task of finding all extensions of theories *color3.board_*n,2,0,0,1,0,0_. Times to find a single extension are short but the list of all solutions is exponentially large.

3.3 Computing with DeReS when no extensions exist

To get a complete picture of the performance of an automated reasoning method (or any search-based algorithm) it is necessary to test the method on instances for which solution do not exist and, hence, the method has to run a complete search. As a benchmark for such tests we use the problem of finding kernels in directed cycles with odd number of vertices. Such graphs have no kernels and, consequently, default theories which we use to encode this problem have no extensions. The objective of this experiment was to find out how long will it take for DeReS to answer that no solution exists. In this case the reasoning time is the same for the query *find one extension* and for the query *find all extensions*.

We use directed cycles C_n of odd length. In Stanford GraphBase C_n is labeled $board(n, 0, 0, 0, 1, 1, 1)$. To encode kernels we use the TheoryBase encoding Δ^3_{ker} and the resulting theories have labels *kernel.board_*n,0,0,0,1,1,1_. In this case, intermediate extensions exist for all strata but the last one. Hence, the entire D must be analyzed before the conclusion that no extensions exist can be drawn. Times needed by DeReS to conclude that no solution exists are shown in Table 5. Since in this case whole set of defaults must be considered to find out that there are no ex-

*kernel.b.board_*$4, m, 0, 0, 5, 3, 1$_ *one solution*								
m	$\|V\|$	$\|E\|$	N	NCPP	CAND	$time_f$	$time_l$	$time_a$
2	8	32	32	52931	2351	8.34	1.28	0.13
3	12	48	48	241868873	9814692	–	–	438.46
4	16	64	64	–	–	–	–	–

Table 3: Searching for a kernel in $board(4, m, 0, 0, 5, 3, 1)$.

*color3.board_*$n, 2, 0, 0, 1, 0, 0$_ *one solution*								
n	$\|V\|$	$\|E\|$	N	NCPP	CAND	$time_f$	$time_l$	$time_a$
200	400	598	2994	7988	2594	1427.50	6.45	0.18
400	800	1198	5994	15988	5194	–	28.72	0.34
600	1200	1798	8994	23988	7794	–	68.74	0.50
800	1600	2398	11994	31988	10394	–	128.17	0.68
1000	2000	2998	14994	39988	12994	–	209.14	0.84

Table 4: Finding a coloring for $L_n = board(n, 2, 0, 0, 1, 0, 0)$.

tensions number of calls to propositional provability procedure is not constant. It grows linearly – 100 additional nodes in the cycle require exactly 3000 extra calls to propositional provability routine. This yields linear reasoning time, $O(N)$, when df-lookup prover is used and quadratic, $O(N^2)$ reasoning time, in the case of local tableau method. The results presented in this section show that relaxed stratification plays a critical role also when determining that no solutions exist.

Also in this case it is important to use an appropriate encoding. Representing the existence of kernel problem for graphs $board(n, 0, 0, 0, 1, 1, 1)$ by theories with the labels *kernel.b.board_*n,0,0,0,1,1,1_ (Δ^1_{ker} in [CMMT95]) leads to a dramatic loss of performance (cf. Table 6).

4 CONCLUSIONS

We have presented several experimental results of using default logic to solve problems from a domain of independent interest – graph theory. In our study we focused on the following two aspects of default reasoning:

1. how the ability to stratify a default theory affects the computational time,
2. which methods of propositional provability are suitable as an oracle for default reasoning.

Consequently, we did not consider any specific method of dealing with a single cluster of defaults. We used a generic reduct-based algorithm to compute extensions for a single stratum of defaults. We expect that relaxed stratification would affect the performance of DeReS in the same way for any other method to process clusters. This claim, however, still needs to be confirmed experimentally.

The results presented in this paper confirmed the theoretical results on relaxed stratification given in [Cho95c]. That is, using relaxed stratification we were able to find extensions for default theories consisting of hundreds and thousands of defaults. Of course, this level of performance is not guaranteed unless the input theory admits a fine stratification, that is, when there is a constant bound on the size of the largest stratum. We showed that relaxed stratification, in general, does not eliminate the exponential time complexity but rather reduces the complexity so that substantial instances can be solved in reasonable time.

We investigated three different methods of implementing provability procedure which is used as an oracle in default reasoning. The three algorithms which we used could have solved a single provability query in amortized times $O(1)$, $O(N)$ and $O(N^2)$ per query. Our results did not depend on any particular algorithm for propositional provability. The same (up to constants) results can be obtained under the assumption that we have three provers which run in constant, linear and quadratic time per query. Two important conclusions are:

1. Performance of DeReS and other nonmonotonic systems can be dramatically improved if local provers are used allowing us to reduce the number of global consistency checks.

kernel.board_n, 0, 0, 0, 1, 1, 1, *compute extensions*								
n	$\|V\|$	$\|E\|$	N	NCPP	CAND	$time_f$	$time_l$	$time_a$
175	175	175	700	5235	2618	414.07	1.46	0.08
375	375	375	1500	11235	5618	4294.84	6.50	0.15
575	575	575	2300	17235	8618	–	15.06	0.24
775	775	775	3100	23235	11618	–	27.25	0.32
975	975	975	3900	29235	14618	–	42.98	0.39

Table 5: Searching for kernels in $C_n = board(n, 0, 0, 0, 1, 1, 1)$.

kernel.b.board_n, 0, 0, 0, 1, 1, 1, *compute extensions*								
n	$\|V\|$	$\|E\|$	N	NCPP	CAND	$time_f$	$time_l$	$time_a$
15	15	15	15	36178	2583	2.53	0.79	0.10
19	19	19	19	318800	17710	30.79	7.27	0.80
23	23	23	23	2670648	121392	334.62	62.49	6.46
27	27	27	27	21633042	832039	3509.69	521.95	51.67
31	31	31	31	171086612	5702886	–	4288.25	403.07

Table 6: Searching for kernels in $C_n = board(n, 0, 0, 0, 1, 1, 1)$.

2. To keep reasoning time as small as possible, the simplest prover which can handle the class of formulas appearing in an input default theory should be used.

Finally, this work demonstrates usability of TheoryBase as a benchmarking system for nonmonotonic reasoning.

Acknowledgments

This work was partially supported by the NSF grant IRI-9400568.

References

[ABW88] K. Apt, H.A. Blair, and A. Walker. Towards a theory of declarative knowledge. In J. Minker, editor, *Foundations of deductive databases and logic programming*, pages 89–142, Los Altos, CA, 1988. Morgan Kaufmann.

[BEP94] R. Ben-Eliyahu and L. Palopoli. Reasoning with minimal models: Efficient algorithms and applications. In *Proceedings of KR'94*, San Francisco, CA, 1994. Morgan Kaufmann.

[Bes89] P. Besnard. *An introduction to default logic*. Springer-Verlag, Berlin, 1989.

[BF91] N. Bidoit and C. Froidevaux. General logical databases and programs: default logic semantics and stratification. *Information and Computation*, 91:15–54, 1991.

[BNN+94] C.Bell, A. Nerode, R.Ng, V.S. Subrahmanian. Mixed Integer Programming Methods for Computing Non-Monotonic Deductive Databases, *Journal of the ACM*, 41:1178–1215, 1994.

[BNN+93] C. Bell, A. Nerode, R. Ng, W. Pugh, and V.S. Subrahmanian. Implementing stable semantics by linear programming. In A. Nerode and L. Pereira, editors, *Logic programming and non-monotonic reasoning*. MIT Press, 1993.

[Bre91] G. Brewka. *Nonmonotonic reasoning: logical foundations of commonsense*. Cambridge University Press, Cambridge, UK, 1991.

[CDS94] M. Cadoli, F. M. Donini, and M. Schaerf. Is intractability of non-monotonic reasoning a real drawback? In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 946–951, Seattle, USA, 1994.

[CH82] A.K. Chandra and D. Harel. Structure and complexity of relational queries.

Journal of Computer and System Sciences 25:99-128, 1982.

[Cho95a] P. Cholewiński. Reasoning with stratified default theories. In *Proceedings of LPNMR'95*. Berlin: Springer-Verlag, 1995. Lecture Notes in Computer Science 928.

[Cho95c] P. Cholewiński. Stratified default theories. In *Proceedings of CSL'94*. Berlin: Springer-Verlag, 1995. Lecture Notes in Computer Science 933.

[CMMT95] P. Cholewiński, W. Marek, A. Mikitiuk, and M. Truszczyński. Experimenting with nonmonotonic reasoning. In *Proceedings of the 12th International Conference on Logic Programming*, Cambridge, MA, MIT Press, 1995.

[Eth88] D. W. Etherington. *Reasoning with incomplete information.* Pitman, London, 1988.

[Gel87] M. Gelfond. On stratified autoepistemic theories. In *Proceedings of AAAI-87*, pages 207–211, Los Altos, CA, 1987. American Association for Artificial Intelligence, Morgan Kaufmann.

[GKPS95] G. Gogic, H. Kautz, Ch. Papadimitriou, and B. Selman. Compactness of knowledge representation: A comparative analysis. In *Proceedings of IJCAI-95*, pages 862–869. Morgan Kaufmann, 1995.

[GL88] M. Gelfond and V. Lifschitz. The stable semantics for logic programs. In R. Kowalski and K. Bowen, editors, *Proceedings of the 5th international symposium on logic programming*, pages 1070–1080, Cambridge, MA, 1988. MIT Press.

[Knu93] D. E. Knuth. *The Stanford GraphBase: a platform for combinatorial computing.* Addison-Wesley, 1993.

[KS89] H.A. Kautz and B. Selman. Hard problems for simple default logics. In *Proceedings of the 1st international conference on principles of knowledge representation and reasoning, KR '89*, pages 189–197, San Mateo, CA, 1989. Morgan Kaufmann.

[McC80] J. McCarthy. Circumscription — a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[MD80] D. McDermott and J. Doyle. Nonmonotonic logic I. *Artificial Intelligence*, 13:41–72, 1980.

[MS72] A.R. Meyer and L.J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential time. In *Proceedings of the 13th annual symposium on switching and automata theory*, pages 125–129. IEEE Computer Society, 1972.

[MT91] W. Marek and M. Truszczyński. Autoepistemic logic. *Journal of the ACM*, 38:588–619, 1991.

[MT93] W. Marek and M. Truszczyński. *Nonmonotonic logics; context-dependent reasoning.* Berlin, Springer-Verlag, 1993.

[MW88] D. Maier and D. S. Warren. *Computing with logic. Logic programming with Prolog.* The Benjamin/Cummings Publishing Company, Inc., 1988.

[Nie95] I. Niemelä. Towards efficient default reasoning. In *Proceedings of IJCAI-95*, pages 312–318. Morgan Kaufmann, 1995.

[NS95] I. Niemelä and P. Simons. Evaluating an algorithm for default reasoning. In *Proceedings of the IJCAI-95 Workshop on Applications and Implementations of Nonmonotonic Reasonings Systems*, 1995.

[Prz88] T.C. Przymusinski. On the declarative semantics of deductive databases and logic programs. In J. Minker, editor, *Foundations of deductive databases and logic programming*, pages 193–216. Los Altos, CA, Morgan Kaufmann, 1988.

[Rei78] R. Reiter. On closed world data bases. In H. Gallaire and J. Minker, editors, *Logic and data bases*, pages 55–76. Plenum Press, 1978.

[Rei80] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[SKC93] B. Selman, H.A. Kautz, and B. Cohen. Local search strategies for satisfiability testing. In *Proceedings 1993 DIMACS*

Workshop on Maximal Clique, Graph Coloring and Satisfiability, 1993.

[SLM92] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of AAAI-92*, pages 440 - 446, Los Altos, CA, 1992. American Association for Artificial Intelligence, Morgan Kaufmann.

[VG88] A. Van Gelder, Negation as failure using tight derivations for general logic programs, in: J. Minker, ed, Foundations of deductive databases and logic programming, pages 1149–1176, Morgan Kaufmann, Los Altos, 1988.

Super Logic Programs

Stefan Brass
Computer Science Dept.
University of Hannover
Lange Laube 22,
D-30159 Hannover, Germany

Jürgen Dix
Computer Science Dept.
University of Koblenz
Rheinau 1,
D-56075 Koblenz, Germany

Teodor C. Przymusinski
Computer Science Dept.
University of California
Riverside,
CA 92521, USA

Abstract

Recently, considerable interest and research effort has been given to the problem of finding a suitable extension of the *logic programming paradigm* beyond the class of normal logic programs. In order to demonstrate that a class of programs can be justifiably called an *extension of logic programs* one should be able to argue that:

- the proposed *syntax* of such programs resembles the syntax of logic programs but it applies to a significantly broader class of programs;
- the proposed *semantics* of such programs constitutes an intuitively natural extension of the semantics of normal logic programs;
- there exists a reasonably simple procedural mechanism allowing, at least in principle, to compute the semantics;
- the proposed class of programs and their semantics is a special case of a more general non-monotonic formalism which clearly links it to other well-established non-monotonic formalisms.

In this paper we propose a specific class of extended logic programs which will be (modestly) called *super logic programs* or just *super-programs*. We will argue that the class of super-programs satisfies all of the above conditions, and, in addition, is sufficiently flexible to allow various application-dependent extensions and modifications. We also provide a brief description of a Prolog implementation of a *query-answering interpreter* for the class of super-programs which is available via FTP and WWW.

1 Introduction

Recently, considerable interest and research effort[1] has been given to the problem of finding a suitable extension of the *logic programming paradigm* beyond the class of normal logic programs. In particular, considerable work has been devoted to the problem of defining natural extensions of logic programming that ensure a proper treatment of *disjunctive information*. However, the problem of finding a suitable semantics for disjunctive programs and databases proved to be far more complex than it is in the case of normal, non-disjunctive programs[2].

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:

- *Null values:* 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 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

[1] It suffices just to mention several recent workshops on *Extensions of Logic Programming* specifically devoted to this subject ([DPP95, DHSH96]).

[2] 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, Min93].

represented most naturally by multiple intended models;

- *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 significantly simplifies the problem of *translation* of non-monotonic formalisms into logic programs, and, consequently, facilitates using logic programming as an *inference engine* 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.

What then should be viewed as an "extension of logic programming"? We believe that in order to demonstrate that a class of programs can be justifiably called an *extension of logic programs* one should be able to argue that:

- the proposed *syntax* of such programs resembles the syntax of logic programs but it applies to a significantly broader class of programs, which includes the class of disjunctive logic programs as well as the class of logic programs with "classical" (or strong) negation;
- the proposed *semantics* of such programs constitutes an intuitively natural extension of the semantics of normal logic programs, which, when restricted to normal logic programs, coincides with one of the well-established semantics of normal logic programs;
- there exists a reasonably simple procedural mechanism allowing, at least in principle, to compute the semantics[3].

It would be also desirable for the proposed class of programs and their semantics to be a special case of a more general non-monotonic formalism which would clearly link it to other well-established non-monotonic formalisms.

[3]Observe that while such a mechanism cannot even in principle be efficient, due to the inherent NP-completeness of the problem of computing answers to just positive disjunctive programs, it can be efficient when restricted to specific subclasses of programs and queries and it can allow efficient approximation methods for broader classes of programs.

In this paper we propose a specific class of such extended logic programs which will be (modestly) called *super logic programs* or just *super-programs*. We will argue that the class of super-programs satisfies all of the above conditions, and, in addition, is sufficiently flexible to allow various application-dependent extensions and modifications. We also provide a brief description of a Prolog implementation of a *query-answering interpreter* for the class of super-programs which is available via FTP and WWW[4].

The paper is organized as follows. In the next Section 2 we recall the definition of a general and highly expressive non-monotonic formalism, introduced earlier by Przymusinski in [Prz95a], by introducing the class of *non-monotonic knowledge databases*. Subsequently, we establish a new and powerful fixed-point characterization of the semantics of such knowledge databases, which provides the foundation for the remaining results obtained in the paper. In Section 3 we introduce the class of super logic programs as a special subclass of the class of all non-monotonic knowledge databases and we establish basic properties of super programs. In the following Sections 4 and 5 we describe two important characterizations, one of which is syntactic and the other model-theoretic, of the semantics of super-programs which, due to the restricted nature of super programs, are significantly simpler than those applicable to arbitrary non-monotonic knowledge databases. Finally, in Section 6 we briefly describe our implementation of a *query-answering interpreter* for super-programs which is based on the previously established model-theoretic characterization of their semantics. We conclude with some final remarks in Section 7.

2 Non-Monotonic Knowledge Databases

We first define the notion of a non-monotonic knowledge database. Super logic programs, defined in the next section, are special knowledge databases.

Consider a fixed propositional language $\mathcal{L}$ with standard connectives ($\vee$, $\wedge$, $\rightarrow$, $\neg$) and the propositional letters *true* and *false*. We denote the set of its propositions by $At_{\mathcal{L}}$. Extend the language $\mathcal{L}$ to a propositional modal language $\mathcal{L}_{Not}$ by augmenting it with a modal operator *Not*, called the *default negation* operator. The atomic formulae of the form *Not* F, where F is an arbitrary formula of $\mathcal{L}_{Not}$, are called *default negation atoms* or just *default atoms*. The formulae of the

[4]See `ftp://ftp.informatik.uni-hannover.de/-software/static/static.html`.

original language $\mathcal{L}$ are called *objective*. Any propositional theory in the modal language $\mathcal{L}_{Not}$ will be called a *non-monotonic knowledge database*. Observe that arbitrarily deep level of *nested default negations* is allowed in the language. For the sake of simplicity, in this paper we consider only *finite* theories.

We assume the following two simple axiom schemata and one inference rule describing the arguably obvious properties of default atoms:

(K') Distributive Axiom:
For any formulae F and G:

$$Not\,(F \vee G) \leftrightarrow Not\,F \wedge Not\,G \qquad (1)$$

(D) Consistency Axiom:

$$Not\,(false) \quad \text{and} \quad \neg Not\,(true) \qquad (2)$$

(N') Invariance Inference Rule:
For any formulae F and G:

$$\frac{F \leftrightarrow G}{Not\,F \leftrightarrow Not\,G} \qquad (3)$$

The first axiom (K') states that default negation *Not* is distributive with respect to disjunctions. The consistency axiom (D) states that *false* is assumed to be false by default but *true* is not. The invariance inference rule (N') states that if two formulae are known to be equivalent then so are their default negations. In other words, the meaning of $Not\,F$ does not depend on the specific form of the formula F, e.g., the formula $Not\,(F \wedge \neg F)$ is equivalent to $Not\,(false)$ and thus is true by (D).

Definition 2.1 (Derivable Formulae)
For any knowledge database T we denote by $Cn_{Not}(T)$ the smallest set of formulae of the language $\mathcal{L}_{Not}$ which contains T, all the (substitution instances of) the axioms (K') and (D) and is closed under both propositional consequence and the invariance rule (N').

We say that a formula F is derivable *from the knowledge database T if F belongs to $Cn_{Not}(T)$. A knowledge database T is* consistent *if $Cn_{Not}(T)$ is consistent.*

The notion of a non-monotonic knowledge database defined above is equivalent to the notion of a *belief theory* in the *Autoepistemic Logic of Beliefs*, *AELB*, originally defined by Przymusinski in [Prz95a]. Although a slightly different notation is used, one can easily see that after replacing everywhere $Not\,F$ by $\mathcal{B}\neg F$ the axioms (K'), (D) and (N') introduced above become equivalent to the axioms (K), (D) and (N) originally used in [Prz95a]. However, the definition given in here is better suited for the introduction of the class of super logic programs.

As the name indicates, non-monotonic knowledge databases must be equipped with a *non-monotonic semantics*. Intuitively this means that we need to provide a meaning to the default negation atoms $Not\,F$. We want the intended meaning of default atoms $Not\,F$ to be based on the principle of *predicate minimization*:

$$Not\,F \text{ holds if } \neg F \text{ is minimally entailed}$$
$$\text{iff}$$
$$F \text{ is false in all minimal models.}$$

In order to make this intended meaning precise we first have to define what we mean by a minimal model of a knowledge database.

Definition 2.2 (Minimal Models) *[Prz95a]*
By a minimal model *of a knowledge database 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 default atoms $Not\,F$. 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[5] *by T.*

Like it is the case with Clark's completion of a logic program P, the non-monotonic semantics of a knowledge database T is given by defining its *static completion* $\overline{T}$.

Definition 2.3 (Static Completion)
The static completion $\overline{T}$ of a knowledge database T is defined as:

$$\overline{T} = Cn_{Not}(T \cup \{Not\,F : T \models_{\min} \neg F\}),$$

where the F's range over all formulae of $\mathcal{L}_{Not}$. The static semantics *of a knowledge database T is defined as the set of all formulae in its completion $\overline{T}$ and thus it coincides with $\overline{T}$.*

Static completion (or static semantics) of a knowledge base T is obtained therefore by: (1) adding to T all default atoms $Not\,F$ satisfying the condition that the formula $\neg F$ is minimally entailed by T, and, (2) closing the resulting theory under the consequence operator

[5]For readers familiar with *circumscription*, this means that we are considering circumscription $CIRC(T;\mathcal{L})$ of the knowledge database T in which objective atoms are minimized while the default atoms $Not\,F$ 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 default atoms and then *minimizing* objective atoms (see also [YY93]).

Cn_{Not}. Consequently, the definition of static completions *enforces* the intended meaning of default atoms described above.

The above definition of static completion is very simple and straightforward and it does *not* involve any fixed-point computation. The main result of this paper, which provides the foundation for all the remaining results, proves that static completion can be equivalently defined as the *least fixed point* of a natural operator defined below. In particular, it shows that the notion of a static completion given above coincides with the notion of the *least static expansion* in *AELB* mentioned above. The existence of an equivalent non-fixed point definition of a static completion significantly simplifies this notion and the underlying theory. First, we consider a very simple example.

Example 2.4 Consider a knowledge database T given by $Not\, a \rightarrow b$. It is easy to see that $T \models_{min} \neg a$ because a is false in all minimal models of T. As a result, the static completion $\overline{T}$ of T contains $Not\, a$. Consequently, as expected, the static semantics of T derives b.

Let T be a fixed knowledge database. We now define (see [Prz95a]) the monotonic[6] *default closure* operator Ψ_T which transforms any database S into its default closure:

$$\Psi_T(S) = Cn_{Not}(T \cup \{Not\, F : S \models_{\min} \neg F\}),$$

obtained by (1) adding to T all default atoms $Not\, F$ satisfying the condition that the formula $\neg F$ is minimally entailed by S, and, (2) closing the resulting theory under the consequence operator Cn_{Not}. The F's range over all formulae of $\mathcal{L}_{Not}$.

Any fixed point of the operator Ψ_T is called a static expansion of T:

Definition 2.5 (Static Expansion) *[Prz95a]*
A knowledge database T° is called a static expansion *of a knowledge database T if it is a fixed-point of the operator Ψ_T:*

$$T^\circ = \Psi_T(T^\circ) =_{def} Cn_{Not}(T \cup \{Not\, F : T^\circ \models_{\min} \neg F\}),$$

where F ranges over all formulae of $\mathcal{L}_{Not}$.

Clearly, the static completion $\overline{T}$ of T can be equivalently defined as $\overline{T} = \Psi_T(T)$. In other words, $\overline{T}$ is obtained as a result of a *single iteration* of the operator Ψ_T.

[6]Strictly speaking the operator is only monotone on the lattice of all theories of the form $T \cup M_{Not}$ where T is fixed and M_{Not} ranges over all sets of default negation atoms.

It follows from the results obtained in [Prz95a] that every database T has the *least* (in the sense of set-theoretic inclusion) static expansion T°, i.e., the least fixed point of the operator Ψ_T. However, the following powerful and yet surprising result shows that for every database T its least static expansion is obtained by a *single iteration* of the operator Ψ_T and thus it always coincides with the static completion $\overline{T}$ of T.

Theorem 2.6 (Main)
Let T be any knowledge database. The least static expansion of T is always obtained by a single iteration of the operator Ψ_T. In other words, the least fixed point of the operator Ψ_T coincides with the static completion $\overline{T}$ of T:

$$\overline{T} =_{def} \Psi_T(T) = \Psi_T(\overline{T}).$$

Since static completion is the least static expansion, a knowledge database T either has a *consistent* completion $\overline{T}$ or it does *not* have any consistent static expansions at all.

Corollary 2.7 *The notion of a static completion given above coincides with the notion of the* least static expansion *in the Autoepistemic Logic of Beliefs,* AELB, *defined in [Prz95a].*

Before presenting the next simple example, we will need the following technical lemma. Its first statement says that if a formula F is assumed to be false by default then its negation $\neg F$ is not.

Lemma 2.8 *If $T = Cn_{Not}(T)$ then for any two formulae F and G of $\mathcal{L}_{Not}$:*

$$T \models Not\, F \rightarrow \neg Not\, \neg F$$
$$T \models (Not\, F \leftrightarrow Not\, G) \leftrightarrow Not\, \neg(\neg F \leftrightarrow \neg G).$$

Example 2.9 Consider a knowledge database T:

$$Not\, a \rightarrow b \quad (4)$$
$$Not\, b \rightarrow c. \quad (5)$$

Obviously $T \models_{min} \neg a$ and $T \models_{min} \neg Not\, b \leftrightarrow \neg c$. Thus the static completion $\overline{T}$ contains $Not\, a$, b and $Not\, \neg(\neg Not\, b \leftrightarrow \neg c)$. By Lemma 2.8, $\overline{T}$ also contains $Not\, Not\, b \leftrightarrow Not\, c$. Moreover, $\overline{T}$ contains $Not\, \neg b$ and thus, by Lemma 2.8, $\neg Not\, b$. Consequently, $\overline{T}$ contains $Not\, Not\, b$ and thus also $Not\, c$. As expected, the static semantics of T derives $Not\, a$, b, $Not\, c$.

3 Super Logic Programs

We now introduce the class of *super logic programs*, briefly called *super programs*, as a subclass of the

class of all non-monotonic knowledge databases. This powerful class of databases includes all *propositional theories*, all *disjunctive logic programs* and all *extended logic programs* with "classical" or strong negation [GL90, APP96].

Definition 3.1 (Super Logic Programs) *A Super Logic Program is a non-monotonic knowledge database consisting of super-clauses of the form:*

$$F \leftarrow G \wedge Not\, H$$

where F and G are arbitrary objective *formulae and H is an arbitrary* positive[7] *objective formula.*

Observe that since any formula $F \leftarrow G$ can be written as a finite conjunction of disjunctions of literals, we can always replace a super-clause by a finite set of clauses whose heads are disjunctions $A_1 \vee \ldots \vee A_k$ of atoms and whose positive premises are conjunctions $B_1 \wedge \ldots \wedge B_m$ of objective atoms. Finally, since the operator *Not* obeys the *distributive law for disjunction* (K'), the default atom $Not\, H$ can be replaced by the conjunction $Not\, C_1 \wedge \ldots \wedge Not\, C_n$ of default atoms $Not\, C_i$, where each of the C_i's is a conjunction of objective atoms. This allows us to establish the following useful fact.

Proposition 3.2 *A super logic program P can be equivalently defined as a set of clauses of the form:*

$$A_1 \vee \ldots \vee A_k \leftarrow B_1 \wedge \ldots \wedge B_m \wedge Not\, C_1 \wedge \ldots \wedge Not\, C_n,$$

where A_i's and B_i's are objective atoms and C_i's are conjunctions of objective atoms.

If $k \neq 0$, for all clauses of P, then the program is called affirmative.

Clearly the class of super-programs contains all propositional theories and is syntactically significantly broader than the class of normal logic programs. In fact, it is much broader than the class of programs usually referred to as *disjunctive logic programs* because:

- It allows constraints, i.e., headless rules. In particular, it allows the addition of *strong* or *"classical"* negation to such programs by introducing *new objective atoms*, say $\neg A$, and assuming the *strong negation axioms* [APP96] $\leftarrow A \wedge \neg A$, which themselves are program rules (rather than meta-level constraints).
- It allows premises of the form $Not\, C$, where C is not just an atom but a conjunction of atoms. This proves very important when reasoning with default assumptions which themselves are *rules*, such as $Not(man \wedge \neg human)$, which can be interpreted as stating that we can assume by default that every man is a human (note that $\neg human$ represents strong negation of *human*).

Although the class of super programs is quite broad, it is significantly less general than the class of all non-monotonic knowledge databases because it *does not allow any nesting of default negation*. As a result, static completions of super programs admit much simpler and computationally more feasible characterizations that allow the computation of their relevant parts without the need to involve arbitrarily deeply nested default negations, which are inherently *infinite* even for finite programs.

In the following two sections we provide two such powerful characterizations of static completions of super programs and one of them is later used to implement a query-answering interpreter for super programs. First, however, we discuss general properties of super programs and give a simple example.

The class of super programs has a number of useful properties. Our first result shows that for any formula F the static semantics $\overline{P}$ of a super program P derives $Not\, F$ if and only if $\neg F$ is minimally entailed by $\overline{P}$. This further clarifies the meaning of negation by default.

Theorem 3.3 *Let P be any super logic program and let $\overline{P}$ be its static completion. For any formula F, $\overline{P} \models Not\, F \equiv \overline{P} \models_{min} \neg F$.*

The next result follows from Corollary 2.7 and the results established in [Prz95a].

Theorem 3.4 (Basic Properties)

- *Every affirmative super program has a consistent static completion. In particular, this applies to all disjunctive logic programs.*
- *For any normal logic program static semantics coincides with the well-founded semantics. Moreover, there is a one-to-one correspondence between consistent static expansions of a normal program and its partial stable models. Under this correspondence, (total) stable models correspond to those consistent static expansions which satisfy the axiom $Not\, A \vee Not\, \neg A$, for every objective atom A.*

[7] I.e., a formula that can be represented as a disjunction of conjunctions of atoms.

- *For any positive disjunctive logic program static semantics coincides with the minimal model semantics.*

Example 3.5 Consider a simple super-program P:

$$\begin{aligned} VisitEurope \vee VisitAustralia \\ Happy \leftarrow VisitEurope \vee VisitAustralia \\ Bankrupt \leftarrow VisitEurope \wedge VisitAustralia \\ Prudent \leftarrow Not\,(VisitEurope \wedge VisitAustralia) \\ Disappoint. \leftarrow Not\,(VisitEurope \vee VisitAustralia). \end{aligned}$$

Given the usual exclusive interpretation of disjunctions we would most likely conclude that we will visit either Australia or Europe but not both and thus we are likely to expect the answer to the query $VisitEurope \vee VisitAustralia$ to be positive resulting in a positive answer to the query $Happy$. Similarly, we expect the answer to the query $Not\,(VisitEurope \wedge VisitAustralia)$ to be positive resulting in a positive answer to the query $Prudent$.

On the other hand, we expect the answer to the query $VisitEurope \wedge VisitAustralia$ to be negative thus resulting in a negative answer to $Bankrupt$. Similarly, we expect the answer to the query $Not\,(VisitEurope \vee VisitAustralia)$, or, equivalently $Not\,VisitEurope \wedge Not\,VisitAustralia$ to be negative thus resulting in a negative answer to the query $Disappoint.$.

The query $Not\,(VisitEurope \wedge VisitAustralia)$ intuitively means *"can it be assumed by default that we don't visit both Europe and Australia?"* and is different from the query $Not\,VisitEurope \wedge Not\,VisitAustralia$ which says *"can it be assumed by default that we don't visit Europe and can it be assumed by default that we don't visit Australia?"*.

It turns out that static semantics produces precisely the intended meaning discussed above. Indeed, clearly $Happy$ must belong to the static completion of P. It is easy to check that $\neg VisitEurope \vee \neg VisitAustralia$ holds in all minimal models of the program and therefore $Not\,(VisitEurope \wedge VisitAustralia)$ must be true in the completion. This proves that $Prudent$ must hold in the completion. Moreover, since $Bankrupt$ is false in all minimal models $Not\,Bankrupt$ must also belong to the completion. Finally, since $VisitEurope \vee VisitAustralia$ is true we infer from (D), (N') that $\neg Not\,(VisitEurope \vee VisitAustralia)$ holds and thus $\neg Disappoint.$ is minimally entailed which implies that $Not\,Disappoint.$ belongs to the completion.

4 Simple Fixed Point Characterization of Static Semantics

Due to Proposition 3.2, we can consider the language of super logic programs to be restricted to the subset:

$$\mathcal{L}^*_{Not} = \mathcal{L} \cup \{Not\,E : E \text{ is a conj. of objective atoms}\}$$

of the full language $\mathcal{L}_{Not}$. In particular, the language $\mathcal{L}^*_{Not}$ does *not* allow any *nesting* of default negations. In fact, default negation can be only applied to conjunctions of objective atoms.

Clearly, in order to answer queries about a super program P we only need to know which formulae of the restricted language $\mathcal{L}^*_{Not}$ belong to the static completion of P. In other words, we only need to compute the restriction $\overline{P}|\mathcal{L}^*_{Not}$ of the static completion $\overline{P}$ to the language $\mathcal{L}^*_{Not}$. It would be nice and computationally a lot more feasible to be able to compute this restriction without having to first compute the full completion, which involves arbitrarily deeply nested default negations and thus is inherently *infinite* even for finite programs. The following important result provides a positive solution to this problem in the form of a much simplified syntactic fixed point characterization of static completions. In the next section we provide yet another solution in the form of a model-theoretic characterization of static completions.

Theorem 4.1 (Fixed-Point Characterization) *The static completion $\overline{P}$ of a super program P can be constructed as follows. Let $P^0 = P$ and suppose that P^n has already been defined for some natural number n. Define P^{n+1} as follows:*

$$Cn(\, P \cup \{Not\,E_1 \wedge \ldots \wedge Not\,E_n \rightarrow Not\,E_0 : P^n \models_{\min} \neg E_1 \wedge \ldots \wedge \neg E_n \rightarrow \neg E_0\}\,),$$

where E_i's range over all conjunctions of objective atoms and Cn denotes the standard *propositional consequence operator. We allow the special case that E_0 is the empty conjunction (i.e. true) and identify Not true with false.*

The sequence $\{P^n\}$ is monotonically increasing and has a unique fixed point $P^m = P^{m+1}$, for some natural number m. Moreover, $P^m = \overline{P}|\mathcal{L}^*_{Not}$, i.e., P^m is the restriction of the static completion of P to the language $\mathcal{L}^*_{Not}$.

The above Theorem represents a considerable simplification over the original definition of static completions given in Definition 2.3. First of all, instead of using the modal consequence operator Cn_{Not} the definition uses the *standard* propositional consequence operator

Cn. Moreover, instead of ranging over the set of all (arbitrarily deeply nested) formulae of the language $\mathcal{L}_{Not}$ it involves only conjunctions of objective atoms. These two simplifications greatly enhance the implementability of static semantics. On the other hand, however, due to the restriction to the language $\mathcal{L}^*_{Not}$, we can no longer expect the fixed point to be reached in just one step.

5 Model-Theoretic Characterization

In this section, we complement Theorem 4.1 by providing a *model-theoretic* characterization of static completions of super programs. More precisely, we characterize models of static completions restricted to the narrower language $\mathcal{L}^*_{Not}$. The resulting characterization was directly used in a prototype implementation of a query answering interpreter for static semantics described briefly in the next section.

Definition 5.1 (Reduced Interpretations)

1. *Let Obj be the set of all propositional valuations of the objective atoms* $At_{\mathcal{L}}$.
2. *Let Def be the set of all propositional valuations of the default atoms*
$$\{Not\,(p_1 \wedge \cdots \wedge p_n) \mid p_i \in At_{\mathcal{L}}\}.$$
3. *We write interpretation I of the language* $\mathcal{L}^*_{Not}$, *i.e. a valuation of*
$$At_{\mathcal{L}} \cup \{Not\,(p_1 \wedge \cdots \wedge p_n) \mid p_i \in At_{\mathcal{L}}\},$$
as $I = I_{obj} \cup I_{def}$ *with* $I_{obj} \in Obj$ *and* $I_{def} \in Def$.
4. *In order to emphasize the difference, we sometimes call an interpretation* $\mathcal{I}$ *of the complete language* $\mathcal{L}_{Not}$, *i.e. a valuation for* $At_{\mathcal{L}} \cup \{Not\,(F) \mid F \in \mathcal{L}_{Not}\}$, *a full interpretation.*
5. *Given a full interpretation* $\mathcal{I}$, *we call its restriction* $I = I_{obj} \cup I_{def}$ *to the language* $\mathcal{L}^*_{Not}$ *the reduct of* $\mathcal{I}$.

Since an interpretation $I = I_{obj} \cup I_{def}$ assigns truth values to all atoms occurring in a super program, the "is model of" relation between such interpretations and super programs is defined. We call $I = I_{obj} \cup I_{def}$ a minimal model of a super program P if there is no objective interpretation I'_{obj} satisfying

$$\{p \in At_{\mathcal{L}} \mid I'_{obj} \models p\} \subset \{p \in At_{\mathcal{L}} \mid I_{obj} \models p\}$$

such that $I' = I'_{obj} \cup I_{def}$ is also a model of P. This is completely compatible with the corresponding notion for full interpretations.

Now our goal in this section is to characterize the reducts of the full models of the static completion $\overline{P}$ of a super program P. In fact, the objective parts of these reducts are easy, once we have the possible interpretations of the default atoms. The idea how to compute them is closely related to Kripke structures (which is not surprising for a modal logic). The worlds in such a Kripke structure are marked with objective interpretations, and the formula $Not\,(p_1 \wedge \cdots \wedge p_n)$ is true in a world w iff $\neg(p_1 \wedge \cdots \wedge p_n)$ is true in all worlds w' which can be "seen" from w. Due to the consistency axiom, every world w must see at least one world w'. Now all the static semantics does is to ensure that every world w sees only worlds marked with minimal objective models. So given a set $\mathcal{M}_{obj}$ of minimal objective models, an interpretation I_{def} of the default atoms is possible iff there is some subset $\mathcal{M}'_{obj} \subseteq \mathcal{M}_{obj}$ (namely the objective interpretations in the worlds w') such that $I_{def} \models Not\,(p_1 \wedge \cdots \wedge p_n)$ iff $I_{obj} \models \neg(p_1 \wedge \cdots \wedge p_n)$ for all $I_{obj} \in \mathcal{M}'_{obj}$. But the minimal objective models $\mathcal{M}_{obj}$ also depend conversely on the possible interpretations of the default atoms (since a minimal model contains an objective part and a default part). So when we restrict the possible default interpretations, we also get less minimal objective models. This obviously results in a fixpoint computation, formalized by the following operators:

Definition 5.2 (Possible Int. of Default Atoms)

Let P be a super program.

1. *The operator* $\Phi_P: 2^{Def} \to 2^{Obj}$ *yields the minimal objective models given a set of possible interpretations of the default atoms. For every* $\mathcal{M}_{def} \subseteq Def$ *let* $\Phi_P(\mathcal{M}_{def}) :=$
$$\{I_{obj} \in Obj \mid \text{there is a } I_{def} \in \mathcal{M}_{def} \text{ such that } I = I_{obj} \cup I_{def} \text{ is min. model of } P\}.$$
2. *The operator* $\Psi_P: 2^{Obj} \to 2^{Def}$ *yields the possible interpretations of the default atoms, given a set of minimal objective models. For every* $\mathcal{M}_{obj} \subseteq Obj$ *let* $\Psi_P(\mathcal{M}_{obj}) :=$
$$\{I_{def} \in Def \mid \text{there is a non-empty } \mathcal{M}'_{obj} \subseteq \mathcal{M}_{obj} \text{ such that for every } p_1, \ldots, p_n \in At_{\mathcal{L}}: I_{def} \models Not\,(p_1 \wedge \cdots \wedge p_n) \iff \text{for every } I_{obj} \in \mathcal{M}'_{obj}: I_{obj} \models \neg(p_1 \wedge \cdots \wedge p_n)\}.$$

3. *The operator* $\Theta_P: 2^{Def} \to 2^{Def}$ *is the composition* $\Theta_P := \Phi_P \circ \Psi_P$ *of* Φ_P *and* Ψ_P.

Theorem 5.3 (Model-Theor. Characterization)

1. *The operator* Θ_P *is monotone and thus its iteration beginning from the set Def of all interpretations of the default atoms has a fixed point* $Def^\diamond$.
2. $Def^\diamond$ *consists exactly of all* $I_{def} \in Def$ *that are default parts of a full model* $\mathcal{I}$ *of* $\overline{P}$.
3. *A reduced interpretation* $I_{obj} \cup I_{def}$ *is a reduct of a full model* $\mathcal{I}$ *of* $\overline{P}$ *iff* $I_{def} \in Def^\diamond$ *and* $I_{obj} \cup I_{def} \models P$.

Example 5.4 Let us consider the following logic program P:

$$\begin{aligned} p \vee q &\leftarrow Not(\neg r). \\ q &\leftarrow Not(\neg q). \\ r &\leftarrow q. \end{aligned}$$

Since only belief atoms with a single proposition occur in P, it suffices to consider only such belief atoms. We start with the set *Bel* containing all eight possible valuations of the three belief atoms $Not(\neg p)$, $Not(\neg q)$, and $Not(\neg r)$. These can be extended to the minimal reduced models listed in the following table. Note that models number 6 to 10 are equal to models number 1 to 5, except that $Not(\neg p)$ is true in them.

No.	$Not(\neg p)$	$Not(\neg q)$	$Not(\neg r)$	p	q	r
1.	0	0	0	0	0	0
2.	0	0	1	1	0	0
3.	0	0	1	0	1	1
4.	0	1	0	0	1	1
5.	0	1	1	0	1	1
6.	1	0	0	0	0	0
7.	1	0	1	1	0	0
8.	1	0	1	0	1	1
9.	1	1	0	0	1	1
10.	1	1	1	0	1	1

So we see that there are only three possible valuations for the objective parts, i.e. $\Phi_P(Bel)$ is:

p	q	r
0	0	0
1	0	0
0	1	1

Now these minimal objective interpretations are the basis for the next round of belief interpretations. Of course, from every minimal objective interpretation we immediately get a possible belief belief interpretation if we translate the truth of p to the falsity of $Not(\neg p)$.

But we can also combine minimal objective parts conjunctively and let $Not(\neg p)$ be true only if $\neg p$ is true in all elements of some set of minimal models. Thus, we get the following four valuations for the belief atoms in $\Psi_P(\Phi_P(Bel))$:

$Not(\neg p)$	$Not(\neg q)$	$Not(\neg r)$
1	1	1
0	1	1
1	0	0
0	0	0

This means that only the models number $1, 5, 6, 10$ remain possible given the current knowledge about the beliefs.

Now p is false in all of these models, so $Not(\neg p)$ must be assumed, and only the following two valuations of the belief atoms are possible in the fixed point $Bel^\diamond$:

$Not(\neg p)$	$Not(\neg q)$	$Not(\neg r)$
1	1	1
1	0	0

The reduced models of the least static expansion $P^\diamond$ are all reduced interpretations which satisfy the rules of P and have one of these two belief parts.

Taking the reduct of a full interpretation was easy. However, with the following Kripke structure $\mathcal{K} = \langle W, R, V \rangle$ we can extend the reduced interpretations back to full interpretations, which are models of the least static expansion:

- The set of worlds W are the reduced interpretations $I = I_{obj} \cup I_{bel}$ which are models of P and satisfying $I_{bel} \in Bel^\diamond$.
- $R(I, I')$, i.e. I sees I' iff I' is a minimal model of P and for all $p_1, \ldots, p_n \in At_{\mathcal{L}}$:

  $$I \models Not(\neg p_1 \vee \cdots \vee \neg p_n) \implies I' \models \neg p_1 \vee \cdots \vee \neg p_n.$$

- The valuation $V(I)$ of a reduced interpretation $I = I_{obj} \cup I_{bel}$ is the objective part I_{obj}.

Now in order to get from a reduced interpretation $I \in W$ to a full interpretation, we simply assign the formulas $Not(F)$ the truth values they have in the world I.

However, let us finally note that not all full models of the least static expansion are reconstructed in this way, only one representative from every equivalence class with the same reduct. Already the program $P := \{p \leftarrow Not(\neg p)\}$ has infinitely many different full models.

6 Implementation of the Model-Theoretic Characterization

We implemented a *query-answering interpreter for the static semantics* in the class of super programs based on the above model-theoretic characterization[8]. Let us explain in a little more detail how our prototype implementation works. First, because of the large number of possible default atoms, it seems impossible to construct explicitly the set *Def* of all valuations — it would have 2^{2^n} elements, where n is the number of objective atoms. However, it suffices to consider only the default atoms which occur explicitly in the program, other default atoms have no influence on the minimal models and we can always extend a valuation of these "critical" default atoms consistently to a valuation of all default atoms.

Furthermore, it is possible to reduce the number of default atoms occurring in the program by evaluating the simple cases directly. For example, when it is possible to derive the disjunctive fact *VisitEurope* $\vee$ *VisitAustralia* (in the example it is directly given), we can delete a rule like

Disappoint. $\leftarrow$ *Not* (*VisitEurope*)$\wedge$*Not* (*VisitAustralia*),

because we know that the body can never be true (this is a special case of the "negative reduction" transformation used in [BD95]). In this example, deleting the rule also eliminates two default atoms.

So the first step of our algorithm is to compute the residual program and to apply the reduction operators defined in [BD95]. This transformation preserves the minimal models, and often significantly reduces the number of default atoms we have to consider (for instance, no default atoms remain if the input program is stratified and non-disjunctive).

Then we apply the definition of the operators Φ_P and Ψ_P from Section 5 quite literally. However, two points are worth mentioning. In order to compute the minimal models for the Φ_P-operator, we use an extension of Clark's completion. Given an interpretation of the default atoms occurring in the residual program, we evaluate the rule bodies, so we get a set of positive disjunctions $p_1 \vee \cdots \vee p_n$. We compute the completion by treating $p_1 \vee \cdots \vee p_n$ like $p_i \leftarrow \neg p_1 \wedge \cdots \wedge \neg p_{i-1} \wedge \neg p_{i+1} \wedge \cdots \wedge \neg p_n$. Then we use any model generation algorithm.

Next, it would be very inefficient to consider all subsets $\mathcal{M}'_{obj} \subseteq \mathcal{M}_{obj}$, as required in the definition of the Ψ_P-operator. However, in order to check whether a given interpretation I_{def} of the default atoms is selected by Θ_P, only the maximal $\mathcal{M}'_{obj}$ is interesting. We construct it as the set consisting of all $I_{obj} \in \mathcal{M}_{obj}$ satisfying the condition

for every default atom: if $I_{def} \models \mathit{Not}\,(p_1 \wedge \cdots \wedge p_n)$, then $I_{obj} \models \neg(p_1 \wedge \cdots \wedge p_n)$.

Obviously, the inclusion of other I_{obj} into $\mathcal{M}'_{obj}$ would immediately destroy the required property. Next, we check wether the so constructed $\mathcal{M}'_{obj}$ is non-empty. Finally, we check that for every default atom *Not* $(p_1 \wedge \cdots \wedge p_n)$ which is false in I_{def}, there is an $I_{obj} \in \mathcal{M}'_{obj}$ with $I_{obj} \not\models \neg(p_1 \wedge \cdots \wedge p_n)$.

The program prints the fixed point of Θ_P (only the evaluations of the critical default atoms), and then the minimal reduced models the default part of which is contained in Θ°_P. It then waits for queries. We allow *arbitrary queries* to super programs which are answered entirely on the basis of these minimal models.

7 Conclusion

In this paper we introduced the class of super-programs which properly extends the classes of disjunctive logic programs, logic programs with "classical" (or strong) negation and arbitrary propositional theories. We demonstrated that the semantics of super programs constitutes an intuitively natural extension of the semantics of normal logic programs. When restricted to normal logic programs, it coincides with the well-founded semantics, and, more generally, it naturally corresponds to the class of all partial stable models of a normal program.

We proved two powerful syntactic and set-theoretic characterizations of the semantics of super-programs which lead to procedural mechanisms allowing its computation. We used one of these characterizations as a basis for the implementation of a *query-answering interpreter* for super-programs which is available via FTP and WWW. We noted that while no such computational mechanism can be efficient, due to the inherent NP-completeness of the problem of computing answers to just positive disjunctive programs, they can become efficient when restricted to specific subclasses of programs and queries. Moreover, further research is likely to produce more efficient *approximation methods*.

There is an interesting relation of our work to [YY93] where the authors defined the *well-founded circumscriptive* semantics for disjunctive programs (without nested default negations). They introduced the con-

[8] See `ftp://ftp.informatik.uni-hannover.de/-software/static/static.html` from which the interpreter is available via FTP and WWW.

cept of minimal model entailment with *fixed* belief atoms and defined their semantics as a *single step application* of circumscription. Our work can therefore be seen as an extension to theirs.

The class of super programs with static semantics constitutes a special case of a much more expressive nonmonotonic formalism called the *Autoepistemic Logic of Knowledge and Beliefs* , *AELB*, and introduced earlier in [Prz95a]. *AELB* isomorphically includes the well-known non-monotonic formalisms of Moore's *Autoepistemic Logic* and McCarthy's *Circumscription*. Via this embedding, the semantics of super programs is clearly linked to other well-established non-monotonic formalisms.

The proposed semantic framework for super programs is sufficiently flexible to allow various application-dependent extensions and modifications. We have already seen in Theorem 3.4 that by assuming an additional axiom we can produce an extension of the stable semantics instead of the well-founded semantics. By adding the distributive axiom for conjunction we can obtain a semantics that extends the *disjunctive stationary semantics* of logic programs introduced (see [Prz95b]). Many other modifications and extensions are possible including variations of the notion of a minimal model resulting in an *inclusive* instead of *exclusive* interpretation of disjunctions.

Even though all the presented results, as well as the current implementation, are technically limited to finite propositional programs, they can be easily extended to range-restricted disjunctive Datalog programs, i.e., to programs in which every variable in the rule appears also in a positive body literal.

References

[APP96] J. Alferes, L. Pereira, and T. C. Przymusinski. 'classical' negation in non-monotonic reasoning. In *Proceedings of the International Conference on Artificial Intelligence and Mathematics, AI&Math'96, January'96*, pages 1–4, 1996.

[BD95a] Stefan Brass and Jürgen Dix. A General Approach to Bottom-Up Computation of Disjunctive Semantics. In J. Dix, L. Pereira, and T. Przymusinski, editors, *Nonmonotonic Extensions of Logic Programming*, LNAI 927, pages 127–155. Springer, Berlin, 1995.

[BD95] Stefan Brass and Jürgen Dix. Disjunctive Semantics based upon Partial and Bottom-Up Evaluation. In Leon Sterling, editor, *Proceedings of the 12th Int. Conf. on Logic Programming, Tokyo*, pages 199–213. MIT Press, June 1995.

[BD96] Stefan Brass and Jürgen Dix. Characterizations of the Disjunctive Stable Semantics by Partial Evaluation. *Journal of Logic Programming*, forthcoming, 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.

[Bry89] François Bry. Logic programming as constructivism: A formalization and its application to databases. In *Proc. of the Eighth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'89)*, pages 34–50, 1989.

[Bry90] François Bry. Negation in logic programming: A formalization in constructive logic. In Dimitris Karagiannis, editor, *Information Systems and Artificial Intelligence: Integration Aspects*, pages 30–46, Berlin, 1990. Springer.

[DHSH96] Roy Dyckhoff, Heinrich Herre, and Peter Schroeder-Heister, editors. *Extensions of Logic Programming*, LNAI 1050, Berlin, 1996. 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.

[DK89a] P. M. Dung and K. Kanchansut. A fixpoint approach to declarative semantics of logic programs. In E.L. Lusk and R.A. Overbeek, editors, *Proceedings of North American Conference Cleveland,Ohio, USA*. MIT Press, October 1989.

[DK89b] P. M. Dung and K. Kanchansut. A natural semantics for logic programs with negation. In E.L. Lusk and R.A. Overbeek, editors, *Proceedings of the 9th Conference on Foundations of Software Technology and Theoretical Computer Science*, Berlin, 1989. Springer.

[DPP95] J. Dix, L. Pereira, and T. Przymusinski, editors. *Non-Monotonic Extensions of Logic Programming*, LNAI 927, Berlin, 1995. Springer.

[GL90] M. Gelfond and V. Lifschitz. Logic programs with classical negation. In *Proceedings of the Seventh International Logic Programming Conference, Jerusalem, Israel*, pages 579–597, Cambridge, Mass., 1990. Association for Logic Programming, MIT Press.

[Lip79] W. Lipski, Jr. On semantic issues connected with incomplete information databases. *ACM Transactions on Database Systems*, 4:262–296, 1979.

[LMR92] J. Lobo, J. Minker, and A. Rajasekar. *Foundations of Disjunctive Logic Programming*. MIT Press, Cambridge, Massachusetts, 1992.

[Min93] Jack Minker. An Overview of Nonmonotonic Reasoning and Logic Programming. *Journal of Logic Programming, Special Issue*, 17, 1993.

[Prz95a] T. C. Przymusinski. Autoepistemic logic of knowledge and beliefs. (In preparation), University of California at Riverside, 1995. (Extended abstract appeared in 'A knowledge representation framework based on autoepistemic logic of minimal beliefs' In *Proceedings of the Twelfth National Conference on Artificial Intelligence, AAAI-94, Seattle, Washington, August 1994*, pages 952-959, Los Altos, CA, 1994. American Association for Artificial Intelligence, Morgan Kaufmann.).

[Prz95b] Teodor Przymusinski. Semantics of normal and disjunctive logic programs: A unifying framework. In J. Dix, L. Pereira, and T. Przymusinski, editors, *Proceedings of the Workshop on Non-Monotonic Extensions of Logic Programming at the Eleventh International Logic Programming Conference, ICLP'95, Santa Margherita Ligure, Italy, June 1994*, pages 43–67. Springer Verlag, 1995.

[SS95] 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, 1995. Springer.

[YY93] Jia-Huai You and Li-Yan Yuan. Autoepistemic Circumscription and Logic Programming. *Journal of Automated Reasoning*, 10, pages 143–160, 1993.

A Proof of the Main Theorem via Clark's Completion

In this section we present a syntactic characterization of static semantic which shows that the least static expansion of a super-program P can be obtained in just *one step* by closing the theory $P_{comp} = P \cup \{Not\, F : F \in compl(P_{res})\}$ under Cn_{Not}. Here P_{res} denotes the *residuum* of the program P [BD95] and $compl(P_{res})$ denotes (the disjunctive version of) *Clark's completion* of P_{res}. This result not only eliminates the need for multiple iterations in the computation of static expansions of super-programs but it also underscores a very interesting and somewhat intriguing relationship between static semantics and Clark's completion. Since the theory P_{comp} is easily computable, this result reduces reasoning under the static semantics to theorem-proving in the modal logic AELB (either by hand or with an automated theorem prover). In particular, the fixpoint construction of the least static expansion can be altogether avoided.

The idea is to add to P the set of formulae which ensure that models "seen" through the modality *Not* are in fact minimal. An appropriate generalization of Clark's completion accomplishes this, but only for programs *without positive body literals*. Such "residual programs" were introduced and investigated in [Bry89, Bry90, DK89a, DK89b, BD95, BD95a]. Fortunately, it turns out that every program P can be transformed into a residual program P_{res} without changing the set of minimal models. In fact, the two elementary program transformations, "unfolding" (or partial evaluation, GPPE [BD95, SS95]) and "elimination of tautologies", are sufficient to obtain a residual program.

The efficient computation of the residual program and the relation to elementary program transformations is explained in [BD95a]. Here we only give an example. Consider the following program P and its residual program P_{res}:

$$\begin{array}{rcl} & P: & \\ p \vee q & \leftarrow & Not\,(\neg r). \\ q & \leftarrow & Not\,(\neg q). \\ r & \leftarrow & q. \end{array} \qquad \begin{array}{rcl} & P_{res}: & \\ p \vee q & \leftarrow & Not\,(\neg r). \\ q & \leftarrow & Not\,(\neg q). \\ p \vee r & \leftarrow & Not\,(\neg r). \\ r & \leftarrow & Not\,(\neg q). \end{array}$$

The first two rules already have the required form. In the last rule, we unfold the body literal q and obtain the residual program P_{res}. This is in fact nothing else but a form of hyperresolution.

We now define an appropriate version of Clark's completion for residual programs. As usual, the goal is to ensure that an objective atom A is true only if it really has to be true, i.e., if there is a rule with A in the head, in which the body is true and all the other head literals are false.

Definition A.1 (Clark's Completion of P_{res}) *Let P_{res} be a super program without positive body literals. Let $A \in At_{\mathcal{L}}$ be an objective atom and*

$$\begin{array}{c} A \vee B_{1,1} \vee \cdots \vee B_{1,m_1} \leftarrow Not\,(\neg C_{1,1}) \wedge \cdots \wedge Not\,(\neg C_{1,n_1}) \\ \vdots \qquad\qquad \vdots \qquad\qquad\qquad \vdots \\ A \vee B_{k,1} \vee \cdots \vee B_{1,m_k} \leftarrow Not\,(\neg C_{k,1}) \wedge \cdots \wedge Not\,(\neg C_{1,n_k}) \end{array}$$

be all rules in P_{res} with A in the head. Then the completion axiom for A is the formula

$$A \leftrightarrow \bigvee_{i=1}^{k} (\neg B_{i,1} \wedge \cdots \wedge \neg B_{i,m_i} \wedge Not\,(\neg C_{i,1}) \wedge \cdots \wedge Not\,(\neg C_{i,n_i}))$$

As usual, Clark's completion $comp(P_{res})$ of the whole residual program P_{res} is the set of all completion axioms, for every $A \in At_{\mathcal{L}}$. We also use the shorthand

$$Not\,(comp(P_{res})) := \{Not\,(F) \mid F \in comp(P_{res})\}.$$

Lemma A.2 *An interpretation $\mathcal{I}$ is a minimal model of a super program P iff it is a model of $comp(P_{res})$.*

It is interesting to note that when we replace $Not\,(\neg C_{i,j})$ by $\neg C_{i,j}$, we obtain the stable models of P. We used this fact in [BD95a] to compute the stable semantics of disjunctive programs.

Theorem A.3 (Characterization by $comp(P_{res})$) *For every super program P, the least static expansion $P^{\diamond}$ of P is $P^{\diamond} = Cn_{AELB}(P \cup Not\,(comp(P_{res}))$.*

A surprising corollary of the above Theorem is that for super programs a single application of the belief closure operator Ψ_T is sufficient. Indeed, due to Lemma A.2, the completion axioms hold in all minimal models, so they are already believed in the first iteration of Ψ_T.

Decision Theory

Utility Independence in a Qualitative Decision Theory

Fahiem Bacchus
Department of Computer Science
University of Waterloo
Waterloo, Ontario
Canada, N2L 3G1
fbacchus@logos.uwaterloo.ca

Adam J. Grove
NEC Research Institute
4 Independence Way
Princeton NJ 08540, USA
grove@research.nj.nec.com

Abstract

Qualitative accounts of utility modeling and decision theory offer the prospect of reasoning about preference and decision-making without requiring hard-to-obtain numerical probabilities and utilities. It is plausible that such accounts can be found because qualitative criteria (in particular, *dominance*) seems to play a large role in human decision making; the formal quantitative apparatus of *maximum expected utility* tends to be invoked only in the most critical, most finely-balanced, cases.

In this paper, we show how non-probabilistic *independence* concepts—such as preferential independence and utility independence—can be integrated with other sources of quantitative information. It turns out that there are some subtleties involved in making sense of these ideas in a logical framework. The main contribution of this paper is to demonstrate these subtleties, and then give semantics that avoid many of the problems. We then argue that knowledge of utility independence can be a useful addition to the qualitative reasoner's tool-kit.

1 Introduction

It is often suggested that rational decision-making should be based on the principle of *maximum expected utility* (MEU). To use MEU one must have a numeric *utility* function that quantifies how desirable or undesirable each particular state of the world is, and a family of probability distributions over the states of the world. This family of distributions is indexed by the actions one might take, each distribution telling us the probability that any particular state of the world will be brought about by the associated action. The MEU principle advocates selecting that action which leads to the greatest expected utility. For an introduction see, for example, [Fre88, GS88, Sav54].

However, it is well known that there are several epistemological and computational difficulties involved in using MEU. In particular, it is often extremely difficult or even impossible to obtain the probability and utility functions required. People tend to express their beliefs, goals, and preferences in different, generally qualitative, terms and have trouble translating these into numerical distributions and utility functions. Even when it is possible to obtain them, it might not be practical to use numeric probabilities and utilities directly. For instance, if there are n independent Boolean propositions the state space may have size 2^n, so that an explicit listing of probabilities or utilities quickly becomes unmanageable. Furthermore, such a listing might obscure valuable structure or heuristic information that is apparent in a more "natural" specification. For example, we might lose the ability to quickly recognize when dominance arguments render detailed utility calculations redundant.

Such problems are one motivation for recent interest in *qualitative* theories of probability and uncertainty, utility (i.e., preference), and their combination (i.e., decision making). There has been greatest progress on qualitative theories of probability. This includes theories of probabilistic independence (notably Bayes nets [Pea88]) which use qualitative information to simplify the acquisition and use of numerical probabilities, theories of extreme probabilities such as ϵ-semantics [Pea89], and the related area of non-monotonic reasoning [Gin87] which (according to some interpretations) seeks to replace probabilities entirely with a qualitative counterpart.

Work concerning qualitative theories of utilities and decision theory is more recent, and far less developed.[1] Among the papers of direct relevance to us include

[1] However, we acknowledge the large body of existing work on *deontic logics* (e.g., [von51]) and *preference logics* (e.g., [von72]). This appears to be only weakly relevant to our work, however, because these logics do not generally appeal to the expected-utility paradigm.

those by Boutilier [Bou94], Pearl and Tan [Pea93, TP94a, TP94b], and Doyle, Wellman, and Shoham [DW91, DW94, DSW91]. We discuss these and other papers in more detail in Section 3.

In some previous works the approach taken has been to dispense with numeric utility and probability functions, instead replacing them with qualitative analogs. For instance, [Pea93] suggests using probabilities of the form ϵ^k for natural numbers k and ϵ a small positive number, and utilities of the form $\pm(1/\epsilon)^k$. By considering the limit $\epsilon \to 0$ (i.e., where probabilities are "very small", utilities are "very large", and all we care about are order-of-magnitude distinctions) one can hope to simplify the reasoning process. There are many interesting variants of this basic idea, including the use of qualitative probabilities alone (such as κ-rankings [Pea93]) or qualitatively ranked utilities alone [TP94a].

In this paper we also consider the problem of decision-making using qualitative, or limited amounts of quantitative, information. An important contrast with previous work is that we will not assume that probabilities or utilities are themselves qualitative (although they may be). Instead, our goal is to work towards a decision theory that can handle qualitative *knowledge* about probabilities and utilities. The particular focus of this paper is knowledge about *independence* for utilities and preferences. Whether or not the probabilities and utilities about which we make independence assertions are in any sense qualitative is generally an orthogonal issue.[2] In Section 4 we give a result illustrating one way in which these two notions can be usefully combined.

We have investigated independence concepts for preference and utility in recent work [BG95]. Most of these concepts have been known for a long time, albeit perhaps not in the A.I. literature, as part of an area known as *multi-attribute utility theory* [KR76]. In [BG95] we were interested in the possibility of graphical models for these concepts, analogous to graphical techniques for probabilistic independence (such as Bayes Nets). In this paper we suggest a different (although related) use for these concepts: that independence assertions are an important source of information about utilities.[3] Like probabilistic independence, these notions are qualitative, relatively easy for people to access, and can simplify computation. Thus, they can help in achieving a more usable version of (qualitative) decision theory.

There are perhaps two reasons why independence concepts are relatively unexplored in A.I. One, which we will address later in the paper, is simply that they are relatively weak (in contrast to the theory of probabilistic independence, which is mathematically richer). Another is that they have only been defined in a rather simple context, involving product spaces of attributes. In contrast, much of the work in qualitative decision theory uses concepts from logic. Standard multi-attribute utility theory might consider a space described by several attributes including, for example, *health* and *wealth*. The standard theory can make sense of the assertion that, for instance, one's *health* is utility independent of the set of all other attributes (including *wealth*). But the standard formulation would have problems saying 1) one's *health* is utility independent of *wealth simpliciter*, or 2) that the logical sentence *health* $\vee$ *wealth* is independent of everything else, or 3) coping with logical constraints, such that the lowest level of *wealth* is incompatible with the highest level of *health*.

A principal contribution of this paper is to show how to define the standard independence notions in a logically rich context. Although our approach is not technically complex, it has several interesting (and possibly controversial) philosophical aspects, because there are several other definitions one might use. The heart of this paper is Section 3, which presents our proposal and discusses some of the difficulties it tries to address. This section also contains some more detailed comparisons with related work. Section 4 explores further consequences of our definitions. In particular, we state results showing how certain independencies interact with other pieces of qualitative information. For example, there are many cases in which the dominance arguments that one might wish to use are invalid unless appropriate independencies are given. In Section 2 we present the basic background material.

2 Standard Independence Concepts

As we have said, we assume familiarity with the basic ideas and techniques of decision theory and the expected-utility paradigm. In particular, it is outside the scope of this paper to defend the MEU principle or to examine the many alternative decision theories that are occasionally proposed. The purpose of this section is to establish some notation and to then give a brief survey of independence concepts for preference and utility.

We often assume that the set of states S is defined by a set of binary attributes (i.e., propositional variables) $V = \{p_1, \ldots, p_n\}$. Hence, S can be considered to be the set of all truth assignments over V. Furthermore, we can use propositional logic to talk about events over S. In general, it is very useful to also allow non-binary attributes, corresponding to concepts or resources that have more than two levels. Of course, it is trivial to

[2] Note that the phrase "qualitative decision theory" is occasionally used to denote theories in which probabilities and/or utilities are themselves qualitative. This is not our usage here.

[3] Unlike probabilistic independence, there are a number of distinct notions of independence relevant to preference and utility. We will simply use the term "independence" to refer to any of these notions when the distinction is not important.

formulate a "propositional"-style logic which can talk about such attributes as well. Everything in this paper applies whenever all attributes take on a discrete number of values; we will to present our results and discussion in terms of binary-valued attributes, but this is solely for notational simplicity. Continuous or real-valued attributes raise distinct issues, and so will not be considered in this paper.

In the following, if $X \subseteq V$ then $f(X)$ stands for some real valued function all of whose arguments are in X, i.e., $f(X) : 2^X \longrightarrow \mathbb{R}$ is a function that depends on the truth value of the variables in X only. A *utility function*, u, is a function over complete states, i.e., $u(V)$, and thus it can potentially take on exponentially many unrelated values, one for every state.

We will also need to refer to probability distributions over S. More generally, when $X \subseteq V$ and we say that Pr is a probability distribution over X, we mean that Pr is a probability distribution over the set of truth assignments to the variables in X.

A utility function u induces a *preference ordering* $\succeq$ on the probability distributions over S as follows:

$$Pr_1 \succeq Pr_2 \quad \textit{iff} \quad \sum_{s \in S} Pr_1(s)u(s) \geq \sum_{s \in S} Pr_2(s)u(s),$$

where Pr_1 and Pr_2 are two distributions over S. That is, we prefer Pr_1 to Pr_2 if Pr_1 induces greater expected utility. Thus utility serves to characterize not only the agent's values but also its attitudes towards risk: it ranks probabilistic gambles between various outcomes.

Sometimes, instead of considering a preference ordering over probability distributions on states, we are only interested in the order among the states themselves. In particular, note that any utility function u induces a unique preference ordering over the individual states: $s \succeq s'$ for two states s and s' iff $u(s) \leq u(s')$. But the converse is not true: since utility functions also reflect one's attitude towards risk, many distinct utility functions can lead to the same preference ordering over states.[4]

We now briefly summarize a number of standard independence notions for utility and preference. With the exception of the final definition (conditional additive independence [BG95]) these are standard ideas from the field of *multi-attribute decision theory*. This review is based on the following sources [Fis82, Fre88, KR76, KLST71] and our paper [BG95].

The first definition we give is that of *preferential independence*. This is the weakest notion we discuss because it only considers the preference ordering among individual states. Intuitively, a set of attributes X is preferentially independent of everything else, if when we hold everything else fixed (i.e., the values of attributes $V-X$), the induced preference ordering over assignments to X does not depend on the particular values that $V-X$ are fixed to. Thus, we can assert that preferences over X hold *ceteris paribus*—i.e., all else being equal.

Definition 2.1 : Recall that each state $s \in S$ is a truth assignment to the variables in V. If $X \subset V$ we can write s as (α, γ), where α is a truth assignment to the variables in X and γ is a truth assignment to the remaining variables $V - X$.

The set of attributes X is *preferentially independent* of $V-X$ when one's preference order among truth assignments to X does not depend on the particular values that the variables $V-X$ are set to. That is,

$$\forall \gamma, \gamma' \in \textit{"truth assignments over } V - X\textit{"} :$$
$$(\alpha, \gamma) \succeq (\beta, \gamma) \text{ iff } (\alpha, \gamma') \succeq (\beta, \gamma'),$$

where α and β are any two truth assignments to the variables in X. ∎

The concept of *utility independence* is similar, but somewhat stronger because it is concerned with the induced utility function (and not just preferences between individual states). Thus, the relative strength of preference between states (and not just the order of these preferences) must stay the same. Put another way, one's attitude towards risk should not change. Like preferential independence, utility independence can also be viewed as a formalization of *ceteris paribus*.

Definition 2.2: Let $X \subset V$ be some set of attributes, and suppose γ is a truth assignment to the remaining variables $V - X$. Given a probability distribution Pr over X, there is a unique distribution Pr^γ over V such that (1) Pr^γ's marginal over X is Pr, and (2) Pr^γ gives probability 1 to γ.

Given a utility function with associated preference ordering $\succeq$, we define the *conditional preference over X given γ*, $\succeq_\gamma$, to be the preference ordering such that

$$Pr_1 \succeq_\gamma Pr_2 \quad \text{iff} \quad Pr_1^\gamma \succeq Pr_2^\gamma,$$

where Pr_1 and Pr_2 are any two distributions over X. ∎

Definition 2.3 : The set of attributes X is *utility independent* of $V-X$ when conditional preferences over X do not depend on the particular value given to $V-X$. That is,

$$\forall \gamma, \gamma' \in \textit{"truth assignments over } V - X\textit{"} :$$
$$Pr_1 \succeq_\gamma Pr_2 \text{ iff } Pr_1 \succeq_{\gamma'} Pr_2,$$

where Pr_1 and Pr_2 are any two distributions over X. Here $\succeq_\gamma$ and $\succeq_{\gamma'}$ are the conditional preferences over X given γ and γ' respectively. ∎

[4] If u' is a monotonic function of u, then u and u' will lead to the same preference ordering over states. But only if u and u' are linearly related are they equivalent as *utility* functions (i.e., generate the same preferences over probability distributions).

It is worth noting that, if we are just concerned with a single binary attribute being independent of all the others, then utility independence and preferential independence coincide, but this is not the case when the set X contains more than one attribute or if the attributes can take more than one value. In the single-attribute binary case (only), both preferential and utility independence reduce to the particular formalization of *ceteris paribus* given in [DW91].

In general, preferential (resp., utility) independence fails to hold if one has a preference reversal among values of (resp., probabilistic mixtures of values of) the attributes X, when some set of attributes in $V-X$ is changed. Judgments of utility independence and preferential independence appear to be fairly natural and common; see [KR76] for a very extensive discussion. They are, at heart, judgments about *relevance* and people seem to be fairly good at this in general.

We close by considering an even stronger notion: *additive independence.*

Definition 2.4: Let $Z_1, \ldots, Z_k$ be a partition of V. $Z_1, \ldots, Z_k$ are additively independent (for $\succeq$) if, for any probability distributions Pr_1 and Pr_2 that have the same marginals on Z_i for all i, Pr_1 and Pr_2 are indifferent under $\succeq$, i.e., $Pr_1 \succeq Pr_2$ and $Pr_2 \succeq Pr_1$. ∎

In other words, one's preference only depends on the marginal probabilities of the given sets of variables, and not on any correlation between them. Conditional versions of both additive and utility independence can be defined. The definitions require that the specified independence holds whenever some subset of variables are held fixed. For instance, the following independence concept was developed in our earlier work [BG95].

Definition 2.5: X and Y are *conditionally additively independent* given Z (X, Y, Z disjoint, $X \cup Y \cup Z = V$) iff, for any fixed value γ of Z, X and Y are additively independent in the conditional preference structure over $X \cup Y$ given γ. ∎

All of these notions of independence have interesting consequences for the form of the utility function. Although knowing these consequences might help somewhat in understanding our results in Section 4, they are not essential and so we omit them. Just to give a flavor, though, here is one of the strongest and most important:

Proposition 2.6: *([KR76]) $Z_1, \ldots, Z_k$ are additively independent for $\succeq$ iff the utility function representing $\succeq$ can be written as*

$$u(V) = \sum_{i=1}^{k} f(Z_i)$$

for some functions f_i.

3 Independence for Formulas

Most existing research in qualitative decision theory is concerned with assertions about logical formulas. For instance, both [Bou94] and [TP94a] give semantics to the assertions of the form "if ψ is known then φ is preferred to $\neg\varphi$", where φ and ψ can propositional logic formulas. The related area of deontic logic also supposes that one should reason about preference and obligation in a logical setting.

In contrast, the various definitions of independence given in Section 2 only deal with attributes (which for us tend to be individual propositional variables) or sets of attributes, not arbitrary formulas. This is a significant restriction, and the purpose of this section is to show how it can be relaxed.

To see part of the difficulty caused by formulas, first consider the simple case where one variable (p_1 say) is preferentially independent of the remaining variables. To simplify the discussion, suppose that the direction of the preference is towards p_1 being true. Then the definition of preferential independence says that, for every pair of states s, s' that agree on the values given to all the $n-1$ remaining variables, we will always prefer the one in which p_1 is true.

Why is this case so straightforward? There are two distinct reasons. First, it seems to be a reasonable formalization of the idea that p_1 is preferred to $\neg p_1$ *ceteris paribus*, i.e., preferred given that "all else is equal". The point is that there is little doubt as to what "all else" should refer to: we should fix the values of all the propositional variables other than p_1. Second, once we fix the values of the other variables, we are left with only two states: one satisfying p_1 and the other $\neg p_1$. There is no doubt as to what "preference" means here: the former state should have higher utility than the latter.

But now suppose that, instead of a primitive propositional variable, it is an arbitrary logical formula that is "preferred" to its negation. To be concrete, we consider the formula $\varphi = p_1 \otimes p_2$ (i.e., the exclusive-or of p_1 and p_2.) What is the "all else" that we are supposed to hold fixed when comparing φ with $\neg\varphi$? There is no clear answer to this. Furthermore, as we see shortly, once we have fixed "all else", we may be left with more than one φ state and more than one $\neg\varphi$ state. How should we compare them?

Such questions have been considered by Doyle, Shoham, and Wellman [DSW91], and also by Tan and Pearl [TP94b]. Roughly speaking, in the case of $\varphi = p_1 \otimes p_2$ they would fix the values of all propositional variables other than p_1 and p_2. (In general, they fix all propositional variables that are not required to appear in the formula being considered. This is how the *ceteris paribus* condition is interpreted.) Note that, for any fixed values of the other variables, we are left

with a set of four states, corresponding to the four truth assignments to p_1 and p_2. Their interpretation of preference is that, among each such set of four states, the two in which φ is true are preferred to the other two, in which φ is false. (That is, *each* state satisfying φ has higher utility than *both* of the $\neg\varphi$ states.)

In Section 3.1, we argue against this interpretation of preference. Instead, we endorse an interpretation proposed by Jeffrey [Jef65] that is based on the idea of *desirability* or (as we prefer to call it) *conditional expected utility*. We also disagree with the [DSW91, TP94b] interpretation(s) of *ceteris paribus*. In Section 3.2 we present our concerns and give an alternative approach. Our approach uses conditional expected utility as the base semantics for preference and to make sense of the ideas of utility (resp., preferential, additive) independence over arbitrary collections of formulas.

3.1 Conditional Expected Utility

Given that one has a collection of φ states and $\neg\varphi$ states, what does it mean that the former are preferred to the latter? The [DSW91, TP94b] proposal is that all of the former are preferred to all of the latter. This is an extremely strong condition with undesirable consequences.

One important problem is that it becomes impossible to override preferences given more specific information.[5] One cannot say, for instance, that φ is preferred to $\neg\varphi$ and at the same time that, conditioned on some other information ψ, we prefer $\neg\varphi$ to φ. However, the pattern in which a general preference is overridden by its reverse in more specific situations occurs frequently. For example, there is a preference for not having surgery over having surgery, yet in the circumstance where surgery would improve one's long term health this preference might be reversed.[6] Hence, it is essential to be able allow for preference overriding. We note that Tan and Pearl, in [TP94a], acknowledge this and propose a modification to their earlier theory that allows statements about overriding preferences. However, their proposal essentially amounts to the simple stipulation that one should ignore general preferences when they are overridden: the underlying semantics are not changed. This seems unsatisfactory to us. Furthermore, if the underlying semantics is incompatible with such a basic pattern of preference, then one can have little confidence that this is the only problem.

[5] See [TH96] for other criticisms of these semantics for preference.

[6] Note that stating that this preference holds *ceteris paribus* does not address the problem. The assertion that the preference holds *ceteris paribus* still means that it is required to hold under any *fixed* setting of the other conditions. So given the fixed condition of needing surgery, these semantics still force a preference for not having surgery over surgery.

Instead, we prefer Jeffrey's proposal from [Jef65], which we refer to as *conditional expected utility*. This is defined if one has a probability function Pr over the underlying space S. Then the conditional expected utility over any subset $T \subseteq S$ can be defined as

$$U(T) = \frac{\sum_{t \in T} Pr(t)\, u(t)}{Pr(T)} \tag{1}$$

where we use U to denote the aggregate utility function. Thus, if the collection of states satisfying φ has higher conditional expected utility than the collection of states satisfying $\neg\varphi$, then we assert that φ is preferred $\neg\varphi$.

In general, if φ and ψ are arbitrary formulas, then we write $\varphi \succeq \psi$ to assert that $U(\varphi) \geq U(\psi)$, where we identify a formula with the set of states satisfying it. (Similarly, $\varphi \succ \psi$ just if $U(\varphi) > U(\psi)$.) Conditional preferences are also easy to interpret: $\varphi_1 \succ \varphi_2$ *given* ψ means that $U(\varphi_1 \wedge \psi) > U(\varphi_2 \wedge \psi)$. It is easy to see this semantics is compatible with statements involving overridden preferences. For instance, the two statements $\varphi \succ \psi$ and $\psi \wedge \omega \succ \varphi \wedge \omega$ can be consistently asserted together.

Perhaps the best intuitive reading of preferences based on conditional expected utility is that they correspond to how one might react to various pieces of news. If $\varphi \succ \psi$ one should be happier to hear that φ is true than to hear that ψ is true. (In Section 5 we briefly discuss how actions might be introduced.) In this paper, we will not give any further defense of Jeffrey's semantics for utility aggregation, mostly because many of the best arguments are in his book [Jef65].

Another important difference between the notion of $\succeq$ just defined and those proposed in [DSW91, TP94b] is that we have not (as yet) invoked any form of *ceteris paribus* condition. In contrast, as discussed earlier, when [DSW91, TP94b] assert $\varphi \succeq \psi$ they are making an assertion that holds (roughly speaking) when other propositions are fixed, i.e., holds *ceteris paribus*. To express *ceteris paribus* conditions in our context, we provide a general mechanism where by a variety of utility independence assertions can be stated. These assertions can be (but need not be) stated independently of assertions about preference. We present the details of this proposal in the next section.

3.2 Independence and Ceteris Paribus

[DSW91, TP94b] give semantics to preference statements that embeds a notion of *ceteris paribus*. In particular, their interpretation of *ceteris paribus* involves considering fixed values for all the propositional variables not mentioned in the formulas being considered.

One problem is that such semantics are very syntax dependent, and thus the conclusions they support can be rather arbitrary. To see why, consider again the

formula $p_1 \otimes p_2$ (exclusive-or) and suppose that we redefine our vocabulary so that p_1 is replaced by a new propositional symbol p_1', such that $p_1' \equiv p_1 \otimes p_2$. In this new language, the old p_1 would be expressed using a compound sentence; in fact, $p_1 \equiv p_1' \otimes p_2$. Since p_1 and p_1' are interdefinable, the new vocabulary is just as expressive as the old, and so it may only be a matter of convention as to which is used. Yet preferential independence of φ is given two different meanings according to whether p_1 or p_1' is primitive.

An even more important problem is that such semantics are inflexible. These semantics commit to a single, fixed, interpretation of *ceteris paribus* that applies to all assertions about preferences. For instance, these semantics do not easily allow one to say that $p_1 \succeq \neg p_1$ independent of the value of p_2 and p_3, while at the same time allowing this preference to possibly be *dependent* on the value of p_4.

Our proposal, which avoids these problems, depends on the concept of the set of *atoms* formed from a collection of formulas, defined as follows.

Definition 3.1: If Ψ is a set of formulas, the *atoms* of Ψ is the set of all *consistent* conjunctions that can be formed from the members of Ψ by including each $\psi \in \Psi$ or its negation. For example, if $\Psi = \{p, q \wedge r\}$ then the atoms of Ψ are $\{p \wedge (q \wedge r),\ \neg p \wedge (q \wedge r),\ p \wedge \neg(q \wedge r),\ \neg p \wedge \neg(q \wedge r)\}$. ∎

For any k formulas, there will be (at most) 2^k atoms. We say "at most" because all combinations might not be consistent, and in this paper we restrict the term "atom" to logically consistent formulas.

The collection of atoms over any set of formulas can be thought of as a new space of states, in which the given formulas play the role of primitive attributes. Each of these atoms corresponds, in general, to a collection of states from the original state space. From the previous section, we know that it is possible to give any collection of states a "utility" value using the idea of conditional expected utility. Thus, an induced utility function can be defined over the space of atoms. Any assertion of utility independence involving the collection of formulas can now be interpreted as an assertion about this induced utility function. Since the formulas are primitive attributes in the new state space, we can use the standard definitions to interpret these independence assertions.

More formally, let $\Psi = \{\psi_1, \ldots, \psi_k\}$ be a collection of formulas. Let the underlying space be S, with utility function u and probability distribution Pr. Consider the set of atoms of Ψ. Each such atom corresponds to a consistent truth assignment to the formulas in Ψ, where the formula ψ_i is assigned the value true just if it appears positively in the atom. We define a new space S^Ψ consisting of all of these truth assignments. A utility function u^Ψ over S^Ψ is defined using conditional expected utility. Specifically, the utility u^Ψ of a state s in S^Ψ is defined to be the conditional expected utility, in the original space, of the atom that corresponds to s. Similarly, a probability distribution Pr^Ψ over S^Ψ is defined using marginalization. That is, Pr^Ψ of a state s in S^Ψ is the probability under Pr (i.e., in the original space) of the set of worlds satisfying the corresponding atom.

For example, if $\Psi = \{\psi_1, \psi_2, \psi_3\}$ then S^Ψ will be the set of 8 truth assignments to the ψ_i (assuming that all atoms are consistent). Thus, using the above definitions, $u^\Psi(\psi_1 \wedge \neg\psi_2 \wedge \psi_3) = U(\psi_1 \wedge \neg\psi_2 \wedge \psi_3)$. (Recall that U is defined by Equation 1.) Note that here we write the atom itself to refer to the corresponding truth assignment. Similarly, $Pr^\Psi(\psi_1 \wedge \neg\psi_2 \wedge \psi_3) = Pr(\psi_1 \wedge \neg\psi_2 \wedge \psi_3)$.

Using the above correspondences, we interpret assertions of independence among a set of formulas Ψ as making assertions about the utility and probability functions on the induced space S^Ψ. Since, the formulas of Ψ are primitive attributes in the induced space, the standard definitions given in Section 2 can be applied almost without change. The only difference arises because not all possible truth assignments are consistent.

Example 3.2: Suppose that we wish to assert that the set of formulas $\{p, q\}$ is preferentially independent of the formula $p \vee (q \wedge r)$. Then we let $\psi_1 = p$, $\psi_2 = q$, $\psi_3 = p \vee (q \wedge r)$ and $\Psi = \{\psi_1, \psi_2, \psi_3\}$. If we were to ignore the issue of inconsistent atoms and apply Definition 2.1 literally, then this assertion states that the preference ordering among the truth assignments (written as atoms) $\{\neg\psi_1 \wedge \neg\psi_2 \wedge \neg\psi_3,\ \neg\psi_1 \wedge \psi_2 \wedge \neg\psi_3,\ \psi_1 \wedge \neg\psi_2 \wedge \neg\psi_3,\ \psi_1 \wedge \psi_2 \wedge \neg\psi_3\}$ must be the same as that among the truth assignments $\{\neg\psi_1 \wedge \neg\psi_2 \wedge \psi_3,\ \neg\psi_1 \wedge \psi_2 \wedge \psi_3,\ \psi_1 \wedge \neg\psi_2 \wedge \psi_3,\ \psi_1 \wedge \psi_2 \wedge \psi_3\}$. That is, the truth or falsity of ψ_3 should not affect one's preferences between the various valuations of ψ_1 and ψ_2.

However, this is not entirely meaningful: $\psi_1 \wedge \neg\psi_2 \wedge \neg\psi_3$ and $\psi_1 \wedge \psi_2 \wedge \neg\psi_3$ are both inconsistent (since if $\psi_1 = p$ is true then $\psi_3 = p \vee (q \wedge r)$ must be as well), and so are not part of the space S^Ψ. To address this, we weaken the definition of preferential independence slightly to simply require that all induced orderings be consistent with each other. In this example, the preference ordering between $\neg\psi_1 \wedge \neg\psi_2 \wedge \neg\psi_3$ and $\neg\psi_1 \wedge \psi_2 \wedge \neg\psi_3$ must be the same as between $\neg\psi_1 \wedge \neg\psi_2 \wedge \psi_3$ and $\neg\psi_1 \wedge \psi_2 \wedge \psi_3$. However the preferences between the first two atoms and the atoms $\psi_1 \wedge \neg\psi_2 \wedge \psi_3$ and $\psi_1 \wedge \psi_2 \wedge \psi_3$ are not constrained by this assertion. ∎

Definition 3.3: (Preferential independence for formulas.) Let $\Psi = \psi_1, \ldots, \psi_j, \psi_{j+1}, \ldots \psi_k$. The set of formulas $\psi_1, \ldots, \psi_j$ is *preferentially independent* of $\psi_{j+1}, \ldots, \psi_k$ when one's preference order among the truth assignments to $\psi_1, \ldots, \psi_j$ consistent with particular values given to the remaining formulas, does

not depend on the values given to these other formulas.

Formally: For any α, β that are truth assignments to $\psi_1, \ldots, \psi_j$, and γ, γ' that are truth assignments to $\psi_{j+1}, \ldots, \psi_k$, then, if *all* four combinations $\{(\alpha,\gamma), (\alpha,\gamma'), (\beta,\gamma), (\beta,\gamma')\}$ are logically consistent, we must have:

$$(\alpha,\gamma) \succeq (\beta,\gamma) \text{ iff } (\alpha,\gamma') \succeq (\beta,\gamma')$$

where $\succeq$ is the preference relation induced by u^{Ψ}. ∎

The modified definition of utility independence is sufficiently similar in spirit that we do not repeat it here. The definition of additive independence, Definition 2.4, can by applied without any change in wording. Many of the interesting properties of these independence concepts can be shown to carry over to the new definitions. For instance, we note that the analog of Proposition 2.6 still holds.

With these definitions we have the flexibility to make independence assertions entirely separately from statements about the direction of preference. But, as [DSW91, TP94b] have recognized, it is often convenient to be able to assert both together. Suppose we wish to assert, for instance, that φ is utility independent of ψ_1, ψ_2 and that (no matter what particular values we give to ψ_1 and ψ_2) we prefer φ to $\neg\varphi$. To do this, one could assert the utility independence and then state the direction of preference relative to any single (arbitrary) consistent valuation for $\{\psi_1, \psi_2\}$. For instance:

$$\varphi \wedge \psi_1 \wedge \psi_2 \succeq \neg\varphi \wedge \psi_1 \wedge \psi_2,$$

together with the assumption of utility independence, implies that, e.g.,

$$\varphi \wedge \neg\psi_1 \wedge \neg\psi_2 \succeq \neg\varphi \wedge \neg\psi_1 \wedge \neg\psi_2$$

(and similarly for any other consistent valuation for ψ_1 and ψ_2). But it is useful to create a more natural notation for such cases, which avoids the need to choose an arbitrary valuation for ψ_1 and ψ_2. We interpret an expression of the form

$$\varphi_1 \succeq_{\psi_1,\ldots,\psi_k} \varphi_2$$

as asserting (1) that $\{\varphi_1, \varphi_2\}$ is independent of $\{\psi_1, \ldots, \psi_k\}$, and (2) that, conditioned on any fixed consistent valuation of $\{\psi_1, \ldots, \psi_k\}$, φ_1 has higher conditional expected utility than φ_2.

It should be noted that $\varphi_1 \succ_{\psi_1,\ldots,\psi_k} \varphi_2$, does not entail $\varphi_1 \succ \varphi_2$, nor does the converse hold (even in the presence of utility independence). That is, it is possible to partition the state space and assert that $\varphi_1 \succ \varphi_2$ in every member of the partition, yet simultaneously assert that $\varphi_2 \succ \varphi_1$ unconditionally. Consider, for example, the case where $\varphi \succeq \neg\varphi$ *given* ψ and $\varphi \succeq \neg\varphi$ *given* $\neg\psi$. To see why $\varphi \succ \neg\varphi$ need not hold unconditionally, suppose that ψ is a very much more desired alternative to $\neg\psi$ than φ is to $\neg\varphi$, and that ψ and $\neg\varphi$ are strongly correlated, so that when $\neg\varphi$ is true ψ tends to be true also. Then we would much prefer to learn $\neg\varphi$ than φ if this is *all* we learn (hoping of course that ψ is also true). But, if we know the value of ψ (no matter whether we know it to be true of false), we would prefer φ.

If p_1 is a basic proposition, the [DSW91] interpretation of p_1 being preferred to $\neg p_1$ can be written as

$$p_1 \succeq_{p_2,\ldots,p_n} \neg p_1$$

using our notation (where $p_2, \ldots, p_n$ are the rest of the basic propositions). But our proposal is far more general than this, because there is freedom to use other collections of formulas instead of $\{p_2, \ldots, p_n\}$.[7] Indeed, we can assert several comparisons between p_1 and p_2 simultaneously, each relative to a different set of formulas. Of course, as we have noted, we also have the ability to state independence (of various types) independently of any specific preferential comparison. Finally, note that our proposal has no built-in syntax dependence. One can, and must, explicitly decide what formulas are actually relevant to a comparison.

The key to understanding how one can reason with a collection of independence assertions is to realize that assertions of independence involving formulas impose algebraic constraints on both the utilities and the probabilities over the original space.

Example 3.4: Let the basic propositions be p, q, and r. The original space then consists of 8 states, and can be specified by 8 basic probabilities $p_{pqr}, p_{pq\bar{r}}, \ldots, p_{\bar{p}\bar{q}\bar{r}}$ and the 8 basic utilities $u_{pqr}, u_{pq\bar{r}}, \ldots, u_{\bar{p}\bar{q}\bar{r}}$.

Consider the assertion that p is utility independent of $q \wedge r$. According to our semantics, this means that we consider the four atoms of the set $\{p, q \wedge r\}$ (Defn. 3.1). Each atom is attributed a utility as determined by Equation 1. The definition of utility independence reduces in this case to the assertion that that $U(p \wedge (q \wedge r)) - U(\neg p \wedge (q \wedge r))$ have the same sign as $U(p \wedge \neg(q \wedge r)) - U(\neg p \wedge \neg(q \wedge r))$. This is equivalent to the assertion that

$$u_{pqr} - u_{pq\bar{r}}$$

and

$$\frac{p_{p\bar{q}r}u_{p\bar{q}r} + p_{pq\bar{r}}u_{pq\bar{r}} + p_{p\bar{q}\bar{r}}u_{p\bar{q}\bar{r}}}{p_{p\bar{q}r} + p_{pq\bar{r}} + p_{p\bar{q}\bar{r}}} - \frac{p_{\bar{p}\bar{q}r}u_{\bar{p}\bar{q}r} + p_{\bar{p}q\bar{r}}u_{\bar{p}q\bar{r}} + p_{\bar{p}\bar{q}\bar{r}}u_{\bar{p}\bar{q}\bar{r}}}{p_{\bar{p}\bar{q}r} + p_{\bar{p}q\bar{r}} + p_{\bar{p}\bar{q}\bar{r}}}$$

have the same sign. That is, it reduces to an algebraic constraint over the utilities and probabilities of the original space. ∎

[7] We note that [DW94] have a proposal that allows some more flexibility than [DSW91], but it still only allows one interpretation of *ceteris paribus* to apply to any particular set of formulas. Furthermore, their interpretation is built into the semantics of preference assertions, and cannot be modified by assertions in the language they present.

The fact that an assertion about utilities also constrains probabilities may seem surprising, but makes sense philosophically. As we have said, the basic independence concept is *ceteris paribus*. But the condition that "everything else be the same" except for the formula of interest (φ say) is unrealistic. It makes more sense to think of everything else being *as similar as possible* given that φ changes truth value. This phrasing makes the similarity to counterfactual and conditional logic clear (see for instance [Lew73]). In counterfactual logic, for instance, one is interested in what would happen if some assertion were to be true even though it is known to be false. There is general agreement that the appropriate semantics for counterfactuals and conditionals should not consider all the states in which φ is true, but only the most "normal" such states. So we should not be surprised if a robust formalization of *ceteris paribus* should also need a notion of how plausible particular states are. And this is precisely the role of probabilities—to tell us how likely or unlikely we consider various states to be.[8]

Standard independence definitions do not *appear* to be invoking anything other than utilities or preference. However, this is somewhat misleading because information about the similarity of states is hidden in the choice of attributes or *framing* [DW91]. [DW94] discuss this further, and also argue that making sense of *ceteris paribus* requires more structure than just the utilities (unlike us, however, they do not suggest probabilistic semantics). [DSW91] also speculates upon the connection to counterfactual logics, but does not develop the suggestion.

4 Reasoning

4.1 The problem

We suspect that the sound "logic" corresponding to any particular notion of preference is likely to be a weak one. For instance, the sequence of papers [DSW91, DW91, DW94] present various (related) definitions of preference, each of which is, in itself, far stronger than the technique of comparing of conditional expected utility. Yet the associated logics are quite limited. As Doyle, Shoham, and Wellman say in the conclusion of [DSW91]:

> "While the logic displays some intuitive properties ... some common and seemingly natural goal operations are not always valid.
>
> ...
>
> The numerous restrictions ... limit the applicability of the inference rules presented here."

Even among these inference rules, not all are (at least in our opinion) reasonable. For example, in the system of [DSW91], whenever φ logically entails ψ, then each of φ and ψ must be at least as desirable as the other.[9] In other words, their notion of relative desire cannot be used to distinguish between stronger or weaker assertions. This seems very unintuitive to us.

The truth seems to be that there are rather few "logical" laws governing preference which have strong and general intuitive support. The makes it difficult to develop a usefully rich logic for qualitative decision making. We are aware of two responses to this problem. The first approach is that taken by [Bou94, TP94a, TP94b]. These papers augment a rather weak underlying theory with some form of non-monotonic (and hence, unsound) reasoning. For example, [TP94a] are able to draw stronger conclusions by looking at what follows in preferred models that minimize the distinctions between the utilities of states. [Lou90] gives a general discussion and defense of the idea of non-monotonically reasoning about utilities.

Although the idea of using non-monotonic reasoning is surely a promising one, it seems too early to assess its success. One difficulty is that the choice of non-monotonic reasoning system used can appear rather arbitrary. For example, [TP94a] do not provide any extended justification for the definition they present, although there are clearly many alternatives that they could have used instead. Nor we aware of any specific proposal that has been applied to more than one or two examples.

The alternative to non-monotonic reasoning, that we are suggesting, is equally speculative. The idea is that instead of finding a logic for a single definition of preference or desirability, one should consider *all* the diverse sources of qualitative or semi-qualitative information one has—probabilistic independence, logics of likelihood, extreme probabilities, logics of preference and obligation, extreme utilities, independence assertions about utility and preference (the specific contribution of this paper), and more. Even quantitative information should be considered (so long as one is not asked for *all* of the numbers). Our conjecture is that together all these sources of information may enable quite sophisticated reasoning even though (in the absence of non-monotonic reasoning) this may not be the case for any one or two of them alone.

This paper is a step towards supporting this hypothesis. By considering in detail a formalism that allows one to state independencies of various types, we

[8] It might seem that we are exaggerating the connection to counterfactual logic, because semantics for counterfactual logics generally do not use probabilities. However, it is easy to show that standard counterfactual semantics are largely equivalent to certain well-known theories of qualitative probabilities (such as the κ-calculus [Pea93]).

[9] Note that it does not follow from this that their system collapses, because their notion of comparison is not necessarily transitive.

show that such information can support some useful inferences about preference. Nevertheless, in isolation such independence assertions are still not that powerful. Examining combinations of various pieces of information in the context of larger and more realistic problems remains important future work.

In the next section we present a small selection of sound reasoning patterns that take advantage of independence. These results are no more than suggestive of the usefulness of independence assertions in more realistic settings, but they do demonstrate that independence can be used to support some intuitive inferences that one might want to make about preferences. In fact, independence is often needed to ensure that these inferences are sound.

4.2 Some results

Suppose we prefer φ to $\neg\varphi$ and ψ to $\neg\psi$ (i.e., $\varphi \succ \neg\varphi$ and $\psi \succ \neg\psi$ using the semantics for $\succ$ given in Section 3.1). Would we prefer to have them both be true to having just one true, or to them both being false? At first glance one's response might be yes, this seems like a reasonable inference. Yet it is easy to construct counter-examples. Suppose, for instance, that Sue likes John and she also likes Fred. She might prefer to be married to John over not, and also prefer to be married to Fred over not. But at the same time might reasonably prefer to be married to neither over being married to both!

This leads to the obvious (and important) question of when it is in fact legitimate to assert that the combination of preferred goals is preferred. There are presumably many pieces of additional knowledge which could validate such reasoning. As our first result shows, utility independence can sometimes help.

Proposition 4.1: *If $\varphi \succeq \neg\varphi$, $\psi \succeq \neg\psi$, and* either[10] *φ is utility independent of ψ or ψ is utility independent of φ, then $\varphi \wedge \psi \succeq \neg\varphi \wedge \neg\psi$*

Intuitively, if one believes that the direction of preference for (or against) φ, say, would not changed according to the value of ψ, then there is a limit to how undesirable their interaction can be. The result shows that getting two goods is preferable to getting neither *when independence holds*. In the example with Sue it is clear that utility independence does not hold.

"Monotonicity" of preferences is another very important pattern of reasoning. That is, when does $\varphi \wedge \pi \succeq \psi \wedge \pi$ follow from $\varphi \succeq \psi$, where π is another formula? This is actually quite a strong conclusion. One straightforward way of justifying it requires both utility and probabilistic independence.

[10] Utility and preferential independence are not symmetric.

Proposition 4.2: *If $\varphi \succeq \psi$, $\{\varphi, \psi\}$ is utility independent of π, and $\{\varphi, \psi\}$ is probabilistically independent of π,*[11] *then $\varphi \wedge \pi \succeq \psi \wedge \pi$.*

Both independencies are necessary, and utility independence cannot be replaced by preferential independence, if this result is to hold. On the other hand, there are different assumptions that lead to the same conclusion. For instance, suppose we assume that utilities are qualitative, in the sense of [TP94a]. (We omit a formal definition, but the basic idea is that utilities are ordinal ranks, in which maximization replaces addition.[12]) Then with qualitative utilities of this type the assumption of probabilistic independence in the previous result can be dropped.

Proposition 4.3: *If $\varphi \succeq \psi$ and $\{\varphi, \psi\}$ is utility independent of π, then $\varphi \wedge \pi \succeq \psi \wedge \pi$ if utilities are qualitative.*

Additive independence can also be used in conjunction with other knowledge to obtain useful conclusions, as the next result illustrates. Suppose φ is preferred to ψ. Although it might seem intuitive at first that φ alone (i.e., $\varphi \wedge \neg\psi$) should be preferred to ψ alone (i.e., $\neg\varphi \wedge \psi$), a moment's reflection shows that this does not necessarily follow. For example, the news that one didn't win the state lottery (φ) is probably not as upsetting as learning that one's monthly paycheck has been canceled (ψ). But losing the lottery and receiving one's regular pay on time ($\varphi \wedge \neg\psi$) may well be inferior to winning the lottery but losing one's pay ($\neg\varphi \wedge \psi$). If we have additive independence (which may be plausible in this example) and that the second event is less likely than the first (not true in this case), such situations cannot occur:

Proposition 4.4: *If $\varphi \succeq \psi$, $\{\varphi, \psi\}$ is additively independent, and ψ is less probable φ (i.e., $Pr(\psi) \leq Pr(\varphi)$), then $\varphi \wedge \neg\psi \succeq \neg\varphi \wedge \psi$.*

5 Actions and Decisions

In the presentation of this paper we have ignored any explicit discussion of *actions* and decision-making. In principle, what one really wants to do is to consider a family of probability distributions (parameterized by possible actions). Preferential comparisons are usually (but not invariably) of interest because they relate to two or more possible courses of action one might take.

A good response to this concern is given by Jeffrey. As he notes, it is usually possible to treat actions simply as new propositions. Thus, we might have propositional

[11] By which we mean that, for any atom A over $\{\varphi, \psi\}$, $Pr(A|\pi) = Pr(A|\neg\pi) = Pr(A)$.

[12] An alternative but equivalent semantics considers standard utilities of the form $\pm(1/\epsilon)^k$, then considers the limit as $\epsilon \to 0$.

symbols *do-A*, *do-B*, ... that are interpreted as true if and only if the corresponding actions *A*, *B*, ... are being performed. In this fashion, it can be argued, one avoids the need for any special treatment of actions. The decision between *A* and *B* reduces to deciding if *do-A* $\succ$ *do-B*, and if we have a detailed domain theory this may be resolvable within the current framework.

Furthermore, we are advocating that (where possible) knowledge be used that is in the form of qualitative assertions that constrain, but by itself does not fully determine, probability distributions. Such qualitative knowledge may be sufficiently robust that it applies to all possible actions being considered.

However, these responses are incomplete. We believe that the most important extension of the current paper is to investigate the idea of merging the work in this paper with a rich model of action. We believe that this would not require any changes to the basic semantics of preference and independence assertions that we are proposing here. Nevertheless, there remain many details that need to be investigated in order to make the formalism more useful. For example, an approach that might be integrated with the current work is the idea of distinguishing between "controllable" and "uncontrollable" propositions (for instance, as in [Bou94]).

6 Conclusions

In this paper, we have argued that *independence* concepts for utility and preference provides a category of *qualitative* information that can be useful for decision making. This is made more plausible by the analogy with probabilistic independence. By combining Jeffrey's notion of conditional expected utility with definitions from multi-attribute decision theory, we have given formal definitions that allow independence concepts to be used in a very general fashion. Our results show that these concepts can indeed be useful when reasoning about preferences.

Despite these results, one of the conclusions we reach is that the step from a numeric utility function to simple qualitative information about utilities is a large one. No single source or class of qualitative information seems to be that powerful in isolation. We suspect that a strong qualitative decision theory will need to take advantage of many diverse classes of information. Knowledge about independence is one such class, and should not be overlooked.

References

[BG95] F. Bacchus and A. Grove. Graphical models of preference and utility. In *Proceedings 11th Conference on Uncertainty in Artificial Intelligence (UAI 95)*, pages 3–19. Morgan Kaufmann, 1995.

[Bou94] C. Boutilier. Towards a logic of qualitative decision theory. In *Proc. Fourth International Conference on Principles of Knowledge Representation and Reasoning (KR '94)*, pages 75–86, 1994.

[DSW91] J. Doyle, Y. Shoham, and M. P. Wellman. A logic of relative desire (preliminary report). In *Proc. 6th International Symposium on Methedologies for Intelligent Systems*, pages 16–31, 1991.

[DW91] J. Doyle and M. P. Wellman. Preferential semantics for goals. In *Proc. 9th National Conference on Artificial Intelligence (AAAI '91)*, pages 698–703, 1991.

[DW94] J. Doyle and M. P. Wellman. Representing preferences as *ceteris paribus* comparatives. In *AAAI Spring Symposium on decision-theoretic planning*, pages 69–75, 1994.

[Fis82] P. C. Fishburn. *The Foundations of Expected Utility*. Reidel, Dordrecht, 1982.

[Fre88] S. French. *Decision Theory*. Ellis Horwood, Chichester, West Sussex, England, 1988.

[Gin87] M. L. Ginsberg, editor. *Readings in Nonmonotonic Reasoning*. Morgan Kaufmann, San Francisco, CA, 1987.

[GS88] P. Gärdenfors and N. Sahlin, editors. *Decision, Probability, and Utility: Selected Readings*. Cambridge University Press, Cambridge, 1988.

[Jef65] R. C. Jeffrey. *The logic of decision*. University of Chicago Press, 1965.

[KLST71] D. H. Krantz, R. D. Luce, P. Suppes, and A. Tversky. *Foundations of Measurement*. Academic Press, New York, 1971.

[KR76] R. L. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. Wiley and Sons, New York, 1976.

[Lew73] D. Lewis. *Counterfactuals*. Blackwell, 1973.

[Lou90] R. Loui. Defeasible reasoning about utilities and decision trees. In H. Kyburg, R. Loui, and G. Carlson, editors, *Knowledge Representation and Defeasible Reasoning*, pages 345–359. Kluwer, 1990.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[Pea89] Judea Pearl. Probabilistic semantics for nonmonotonic reasoning: A survey. In *Proc. First International Conference on Principles of Knowledge Representation and Reasoning (KR '89)*, pages 505–516, 1989.

[Pea93] J. Pearl. From conditional oughts to qualitative decision theory. In *Proceedings 9th*

Conference on Uncertainty in Artificial Intelligence (UAI 93), pages 12–20. Morgan Kaufmann, 1993. A version of this paper appeared in the 1993 AAAI Spring Symposium Reasoning about Mental States, under the title "A Calculus of Pragmatic Obligation".

[Sav54] L. J. Savage. *The Foundations of Statistics.* Dover, New York, 1954.

[TH96] R. H. Thomason and J. F. Horty. Nondeterministic action and dominance: Foundations for planning and qualitative decision. In *Proceedings of the Sixth Conference on Theoretical Aspects of Reasoning about Knowledge (TARK-96)*, pages 229–250, 1996.

[TP94a] S. Tan and J. Pearl. Qualitative decision theory. In *Proc. 12th National Conference on Artificial Intelligence (AAAI '94)*, pages 928–932, 1994.

[TP94b] S. Tan and J. Pearl. Specification and evaluation of preferences under uncertainty. In *Proc. Fourth International Conference on Principles of Knowledge Representation and Reasoning (KR '94)*, pages 530–539, 1994.

[von51] G. H. von Wright. Deontic logic. *Mind*, 60:1–15, 1951.

[von72] G. H. von Wright. The logic of preference reconsidered. *Theory and Decision*, 3:140–167, 1972.

On Stable Social Laws and Qualitative Equilibrium for Risk-Averse Agents

Moshe Tennenholtz
Faculty of Industrial Engineering and Management
Technion–Israel Institute of Technology
Haifa 32000
Israel
e-mail: moshet@ie.technion.ac.il

Abstract

This paper introduces and investigates the notion of qualitative equilibria, or *stable social laws*, in the context of qualitative decision making. Previous work in qualitative decision theory has used the *maximin* decision criterion for modelling qualitative decision making. When several decision-makers share a common environment, a corresponding notion of equilibrium can be defined. This notion can be associated with the concept of a *stable social law*. This paper initiates a basic study of stable social laws; in particular, it discusses the stability benefits one obtains from using social laws rather than simple conventions, the existence of stable social laws under various assumptions, the computation of stable social laws, and the representation of stable social laws in a graph-theoretic framework.

1 Introduction

General coordination mechanisms are essential tools for efficient reasoning in multi-agent AI systems. Coordination mechanisms are a major issue of study in the fields of mathematical economics and game theory as well. Much work in these fields concentrates on the notion of an *equilibrium*. An equilibrium is a joint behavior of agents, where it is irrational for each agent to deviate from that behavior. The notion of an equilibrium discussed in the game theory and mathematical economics literature refers to agents which are expected utility maximizers. However, much work in AI has been concerned with more qualitative forms of rational decision making. In particular, work in AI has been concerned with agents which attempt to maximize their worst case payoff. Although, at first, this behavior may look questionable from a decision-theoretic perspective, it is known to capture the behavior of risk-averse agents (Luce & Raiffa 1957; Dubois & Prade 1995; Brafman & Tennenholtz 1996), and it is appropriate in the context of qualitative decision theory (Boutilier 1994; Tan & Pearl 1994; Dubois & Prade 1995; Darwiche & Goldszmidt 1994). Moreover, in (Brafman & Tennenholtz 1996) Brafman and Tennenholtz have shown general conditions under which an agent can be viewed *as if* it were a *maximin* agent (i.e., an agent which maximizes its worst case payoff). However, the corresponding notion of equilibrium has not yet been investigated. In this paper we introduce this notion and investigate its properties. The concept of qualitative equilibrium for risk-averse agents turns out to coincide with the notion of a *stable social law*, to be introduced later in this paper. For ease of exposition we introduce the notion of a stable social law in a self-contained fashion, as an extension to previous work on *artificial social systems*.

Some work on multi-agent systems assumes that agents are controlled by a single entity which dictates their behavior at each point in time, while some other work is concerned with decentralized systems where no global controller exists. A significant part of the theory developed for decentralized multi-agent systems (Bond & Gasser 1988; Demazeau & Muller 1990) deals with conflict resolution in multi-agent encounters. The basic theme of work on this subject is that in decentralized systems agents will reach states of conflict and appropriate negotiation mechanisms would be needed in order to resolve these conflicts. The result of the negotiation process is a deal that the agents will follow. Work in AI has been mostly concerned with agents that conform to agreed-upon deals. Agents may not follow irrational negotiation protocols, but will conform to deals obtained by following rational negotiation protocols (Kraus & Wilkenfeld 1991; Zlotkin & Rosenschein 1993; Durfee, Lee, & Gmytrasiewicz 1993).[1] This differs from work in game-theory (Owen 1982; Fudenberg & Tirole 1991) where a joint strategy is considered unstable (and therefore unsatisfactory from a design perspective) if an agent has a ratio-

[1] See (Sandholm & Lesser 1995) for a detailed discussion of this point.

nal incentive to deviate from it. The Artificial Social Systems approach (e.g., (Moses & Tennenholtz 1990; Shoham & Tennenholtz 1995)) exposes a spectrum between a totally centralized approach and a totally decentralized approach to coordination. The basic idea of the Artificial Social Systems approach is to add a mechanism, called a social law, that will minimize the need for both centralized control and on-line resolution of conflicts. In a mobile robots setting, for example, such a social law may consist of various traffic constraints (Shoham & Tennenholtz 1992). More generally, a social law is a set of *restrictions* on the agents' activities which allow them enough freedom on one hand, but at the same time constrain them so that they will not interfere with each other. In particular, a social law makes certain conflicts unreachable, and as a result improves the system efficiency. Notice that mechanisms for conflict resolution can serve as part of the social law; they will be used in situations where conflicts can't be prevented in advance.

The motivation for the theory of artificial social systems has been the design of artificial multi-agent systems, and as such it assumes that the agents will obey the law supplied by the designer. However, if each agent is designed by a different designer then some laws might be considered irrational. Therefore, at the current stage, the artificial social systems approach and approaches to conflict resolution are somewhat complementary; the resolution of conflicts in multi-agent encounters is part of a more general theory of social laws, but the theory of artificial social systems has neglected the stability of social laws in multi-agent encounters. In this paper we wish to bridge part of the gap between the theory of artificial social systems and the theory of conflict resolution in multi-agent encounters, by considering *stable social laws* for multi-agent encounters. A social law for a multi-agent encounter is a restriction of the set of available actions (in the encounter) to a set of socially allowed actions. Stable social laws make deviation from them irrational. Notice that a convention is a particular type of a social law; a convention determines a particular joint action for the agents to follow (e.g., keep the right of the road), while a social law allows several such actions and prohibits others. As it turns out, this distinction is quite important and useful in the non-bayesian context frequently adopted in AI.

We will discuss social laws for multi-agent encounters using a game-theoretic framework which is tailored for assumptions made in the AI literature, and especially in recent work on qualitative decision making. In particular, we will assume that the agents are risk-averse agents, which use the *maximin decision criterion*. More specifically, given a set of possible behaviors of the other agents, the aim of an agent is to optimize its worst case outcome assuming the other agents may follow any of these behaviors. This kind of behavior is appropriate where there is some ordinal relation on possible outcomes. In such situations all that matters to agents is the order of payoffs and not their exact value. The precise conditions under which such modelling is an appropriate one are discussed in (Brafman & Tennenholtz 1996). We will require that a social law suggested for a particular encounter will guarantee to the agents a certain payoff, and that it will be stable; there should be no incentive to deviate from it assuming the agents are risk-averse agents. Hence, a stable social law corresponds to a notion of qualitative equilibrium for risk-averse agents.

We start by introducing our framework. In particular, we define the notion of stable social laws. Having the basic framework, we show that the set of multi-agent encounters for which a stable convention exists is a strict subset of the set of multi-agent encounters for which there is an appropriate stable social law; however, we show that there exists situations where no stable social law exists. Then, we initiate a computational study of stable social laws; we formulate the corresponding computational problem and show that the general problem of coming up with a stable social law is intractable; in addition, we point to an interesting restriction on our framework under which the synthesis of stable social laws is polynomial. We then return back to the question of the existence of stable social laws; we first show how this question can be formulated in standard graph-theoretic terms, and then expose a class of encounters where simple graph-theoretic conditions imply the existence of stable social laws. Sketch of proofs can be found in the appendix.

2 The Basic Framework

In this section we introduce our basic framework, which is built upon a basic game-theoretic model.

2.1 The Basic Model

In general AI planning systems, agents are assumed to perform *conditional plans*.[2] A conditional plan is a (perhaps partial) function from the local state of an agent to action. Conditional plans can be treated as *protocols* in distributed systems terms, or as *strategies* in game-theoretic terms. In the sequel we will make use of a game-theoretic model; therefore, we will adopt the term strategy.

Multi-agent encounters can be represented as a game. In this paper we will consider two-person games, where two agents participate in an encounter. We will be concerned with finite games where each agent has a finite number of strategies. A *joint strategy* for the agents consists of a pair of strategies, one for each

[2]Plans with complete information and other forms of plans will be taken as restrictions on the general form of plans considered in this paper; this point will not effect the discussion or results presented in this paper.

agent. Each joint strategy is associated with a certain payoff for each of the agents, as determined by their *utility functions*. The above-mentioned terms are classical game-theoretic terms which capture general multi-agent encounters.

Formally, we have:

Definition 2.1:

A *game* (or a *multi-agent encounter*) is a tuple $\langle N, S, T, U_1, U_2 \rangle$, where $N = \{1, 2\}$ is a set of agents, S and T are the sets of strategies available to agents 1 and 2 respectively, and $U_1 : S \times T \rightarrow \Re$ and $U_2 : S \times T \rightarrow \Re$ are utility functions for agents 1 and 2 respectively.

One interesting point refers to the knowledge of the agents about the structure of the game. In this work, we assume that agents are familiar with the sets of actions available to the different agents, but an agent might be aware only of its own payoff function. Our results are appropriate both for the case where the payoff functions are common-knowledge among the agents and for the case an agent knows only its individual payoff function.

How should agents behave in a multi-agent encounter prescribed by a given game? The system's designer may wish to guarantee to the agents a particular payoff. Methods of negotiation for guaranteeing particular types of efficient behavior are discussed in the Distributed AI literature. Most of this literature assumes that agents do not deviate from agreed-upon joint strategies, although agents may adopt only rational negotiation protocols (Zlotkin & Rosenschein 1993; Kraus & Wilkenfeld 1991). On the other hand, work in Game-Theory has been concerned with finding joint strategies which will be stable against rational deviations, where rationality is associated with expected utility maximization. We are interested in guaranteeing efficient behavior for the agents; this behavior should be stable against rational deviations; however, the notion of rationality we adopt would be different from expected utility maximization and would be in the spirit of qualitative decision theory (Luce & Raiffa 1957; Dubois & Prade 1995; Brafman & Tennenholtz 1996). In this paper we borrow a most classical decision criterion in order to model a rational decision by the agent. Namely, we use the *maximin* decision criterion to be discussed below. The precise conditions under which an agent can be viewed as if it uses this decision criterion are discussed in (Brafman & Tennenholtz 1996). This modeling perspective becomes especially appealing if an agent does not know the payoff function of the other agent.

Definition 2.2: Let S_i be the set of strategies available to agent i, and let u_i be the utility function of agent i. Define $u_1(s, S_2) = min_{t \in S_2} u_1(s, t)$ for $s \in S_1$, and $u_2(S_1, s) = min_{t \in S_1} u_2(t, s)$ for $s \in S_2$. The *maximin value for agent 1 (resp. 2)* is defined by $max_{s \in S_1} u_1(s, S_2)$ (resp. $max_{t \in S_2} u_2(S_1, t)$). A strategy of agent i leading to the corresponding maximin value is called a *maximin strategy for agent i*.

In the sequel, we will assume the agents adopt the maximin decision criterion (i.e., choose a maximin strategy), although many of our results and observations do hold for other qualitative decision criteria as well.

2.2 Conventions and Social Laws

Given a game and a requirement that the agents will be able to obtain a payoff of at least t, the designer may supply the agents with an appropriate convention: a joint strategy for which the utility for both agents is greater than or equals to t. A convention is a special case of a social law. A social law in a multi-agent encounter is a restriction on the set of strategies available to the agents; a convention will simply restrict the behavior to a one particular joint strategy. When selecting a social law the designer may wish to select it in a way which allows each agent at least one strategy which guarantees a payoff of at least t.

Definition 2.3: Given a game $g = \langle N, S, T, U_1, U_2 \rangle$ and an efficiency parameter t, we define a *useful social law* to be a restriction of S to $\bar{S} \subseteq S$, and of T to $\bar{T} \subseteq T$, which satisfies that there exists $s \in \bar{S}$ such that $U_1(s, \bar{T}) \geq t$, and that there exists $k \in \bar{T}$ such that $U_2(\bar{S}, k) \geq t$. A (useful) convention is a (useful) social law where $|\bar{S}| = |\bar{T}| = 1$.

In general, a useful social law is a restriction on each agent's activities which enable each agent to act individually and succeed reasonably well, as long as all the agents conform to the law (see the discussion and the general semantics in (Moses & Tennenholtz 1995; Shoham & Tennenholtz 1995)). At this point the idea of using social laws for coordinating agents' activities in a multi-agent encounter may seem a bit strange; why should we care about social laws if every efficiency degree which can be obtained by a social law can already be obtained by an appropriate simple convention? However, as we will see later, social laws can serve as much more useful entities than simple conventions for agents participating in a multi-agent encounter. We will elaborate on this point later.

3 Stable Social Laws

The concept of a social law which has been discussed in previous work is a general and most powerful tool. In this work we are concerned with social laws for multi-agent encounters. Although that's a most popular setting for the study of the resolution of conflicts, up to date the power of social laws has been illustrated in

more complex settings (Shoham & Tennenholtz 1995; 1992). As we shall see, social laws may serve as useful tools for multi-agent encounters as well.

Definition 3.1: Given a game $g = \langle N, S, T, U_1, U_2 \rangle$ and an efficiency parameter q, a *quasi-stable social law* is a useful social law (with respect to q) which restricts S to $\bar{S}$ and T to $\bar{T}$, and satisfies the following:[3] there is no $s' \in S - \bar{S}$ which satisfies $U_1(s', \bar{T}) > max_{s \in \bar{S}}\{U_1(s, \bar{T})\}$, and there is no $t' \in T - \bar{T}$ which satisfies $U_2(\bar{S}, t') > max_{t \in \bar{T}}\{U_2(\bar{S}, t)\}$.

Hence, a quasi-stable social law will make a deviation from the social law irrational as long as the other agent obeys the law. However, the above definition of stability may not be satisfactory in our context. In a multi-agent encounter an agent has a specific goal to obtain, and there is no reason to assume an agent will execute a strategy which yields to it a payoff which is lower than the payoff guaranteed to it by another strategy, assuming the other agent obeys the law. Putting it in other terms, given that we talk about a specific encounter with specific goals, there is no reason to include in the set of allowed strategies a strategy which is dominated by another strategy in that set. This requirement is consistent with models of stable social situations discussed in the game theory literature (Greenberg 1990).

Formally, we have:

Definition 3.2: A quasi-stable social law is a *stable social law* if the payoff guaranteed to each of the agents is independent of the strategy (conforming to the law) it selects, as long as the other agent conforms to the social law (i.e., selects strategies allowed by the law).

Notice that a stable social law is the equilibrium concept which one may wish to associate with the *maximin* decision criterion. Hence, our study of stable social laws can be interpreted as a basic study of equilibria in the context of qualitative decision making.

In the rest of this paper we discuss stable social laws. Given a multi-agent encounter, a stable social law will guarantee to the agents a particular payoff, similarly to the way a particular payoff can be guaranteed by a simple convention. As it turns out however, the difference between social laws and conventions stems from the fact that social laws may be more stable than conventions in multi-agent encounters. This will be the topic of the following section.

[3] Our definition is in the spirit of classical game-theory; we require that deviation by one agent will be irrational given that the other agent sticks to the suggested behavior.

4 Social Laws vs. Conventions

Having a definition of stable social laws, one may ask: what are these laws good for? If we wish to guarantee a certain payoff for the agents, why can't we look for stable conventions, i.e., select a joint strategy from which a deviation would be irrational, assuming such a strategy exists?

The answer is supplied by the following result:

Theorem 4.1: *There exists games for which there are no stable conventions, but where appropriate stable social laws do exist.*

The above theorem reveals a new contribution of the theory of social laws: restricting the activities of the agents to a set of allowed actions rather than to a particular action is useful even in simple multi-agent encounters. This is due to the fact that social laws may be more stable than simple conventions. The intuition behind the above result is as follows. Although different strategies may lead to similar payoffs, different strategies may block different deviations by the agents. Therefore, the fact that the agent's behavior is only partially defined may improve the system efficiency. Assume for example that there are two agents, each of which can invest its money using four options, A, B, C,or D. If they will invest only in options A and B then they will get reasonable payoffs. However, if they are told to invest in particular options, e.g., one is told to invest in A and the other is told to invest in B, then one of them may take this opportunity in order to gain more on behalf of the other using option C or D. But, if both C and D yield low payoffs when they are applied against A or B (although not against both of them), such deviation can be prevented by telling each agent to choose (non-deterministically) from among options A and B (i.e., by supplying the social law: "don't use C and D", rather than pointing to particular investments). The reader may get additional understanding of this situation by considering the proof of Theorem 4.1.

We have shown that social laws are more stable than conventions. We can also show:

Theorem 4.2: *There exist games for which no stable social laws exist (for any efficiency parameter).*

Hence, stable social laws are powerful but do not always exist. Given this observation, it may of interest to supply a procedure for computing when a social law exists. Naturally, in cases where a stable social law exists it may be of interest to compute such a law. In addition, it may be of interest to characterize conditions for the existence of stable social laws. These are the topics of the following sections.

5 Computing Stable Social Laws

In this section we take a look at the computation of stable social laws. In order to do so, we first need to decide on the representation of our input. We will use the standard game-theoretic representation in which a multi-agent encounter is represented by a game matrix.

The problem of computing a Stable Social Law (SSLP) is defined as follows:

Definition 5.1: The Stable Social Law Problem [SSLP]:

Given a multi-agent encounter g, and an efficiency parameter t, find a Stable Social Law which guarantees to the agents a payoff which is greater than or equals to t if such a law exists, and otherwise announce that no such law exists.

Notice that if we restrict ourselves to simple conventions, the computational problem is easy; however, as we have observed, conventions are not as useful as social laws. As the following Theorem shows this does not come without a cost. We are able to show:

Theorem 5.2: *The SSLP is NP-complete.*

The importance of the above theorem is twofold: first, it supplies an initial result in the computational study of stable social laws. Second, the proof of this result teaches us about the structure of stable social laws. Further understanding of this structure is obtained in the following section, where we supply a graph-theoretic representation of stable social laws.

Given the previous Theorem, it would be of interest to identify general cases where the problem of coming up with a stable social law (if such a law exists) is tractable. One case which is of interest is when the parties involved are of unequal power. One way of capturing this fact is by assuming that one party has much more strategies available to it than the other party does. Formally, we say that an agent is *logarithmically bounded* if the number of strategies available to it is $O(log(n))$ where n is the number of strategies available to the other agent. In this case we can show:

Theorem 5.3: *The SSLP when one of the agents is logarithmically bounded is polynomial.*

6 Graph-Theoretic Representations of Stable Social Laws

We can learn about the structure of stable social laws by studying the reduction in Theorem 5.2. More generally, the study of a new equilibrium concept and of its use can greatly benefit from representation theorems which show what does this concept mean in terms of known concepts. In addition, in the context of this particular work, such representation theorems can supply conditions for the existence of stable social laws. The reduction used in the proof of Theorem 5.2 shows that a *special case* of the problem of finding a stable social law is isomorphic to a well-known problem. This has been useful for proving the above-mentioned result. However, it would be of interest to characterize the general Stable Social Laws concept by means of well-known terminology. In particular, in this section we make use of graph-theoretic terms in order to characterize the stable social law concept.

We will make use of the following standard terms:

Definition 6.1: Let $G = (V, E)$ be a *graph*, where V is a set of *nodes*, and $E \subseteq V^2$ is a set of *edges*. G is *undirected* if, for all $v_1, v_2 \in V$, $(v_1, v_2) \in E$ iff $(v_2, v_1) \in E$, and is *directed* otherwise. A set $V' \subseteq V$ is an *independent set* if there are no $v', v'' \in V'$ which satisfy $(v', v'') \in E$. A set $V' \subseteq V$ is a *clique* if $(v', v'') \in E$ for all $v', v'' \in V'$. A node $v \in V$ is *non-isolated* relative to $V' \subseteq V$ if there is a vertex $v' \in V'$ such that $(v, v') \in E$. A set $V' \subseteq V$ is a *dominating set* if for each node $v' \in V - V'$ there is a node $v'' \in V'$ such that $(v', v'') \in E$. A node $v \in V$ is a *sink* if there is no v' such that $(v, v') \in E$. The graph G is *k-colorable* if we can color the nodes of the graph with k colors in a way that $(v, v') \in E$ implies that v and v' have different colors.

We would now like to make a connection between the above-mentioned graph-theoretic terms and our notion of a stable social law. In the sequel we will be concerned with games where the sets of strategies, S, available to the agents are identical. We will also assume the game is symmetric in the sense that $U_1(s, t) = U_2(t, s)$ (i.e., the outcome of the agents is independent of their names). We will be interested in social laws that are fair, in the sense that if a strategy is prohibited for one agent then it is prohibited for all agents. For ease of exposition we will be concerned with social laws guaranteeing the value t and no more than t.

Definition 6.2: Given a game g and an efficiency parameter t, let $G_1 = (V, E_1), G_2 = (V, E_2), G_3 = (V, E_3)$ be directed graphs where V is associated with the set of strategies S, and E_i is defined as follows: $(s, q) \in E_1$ iff $U_1(s, q) \geq t$; $(s, q) \in E_2$ iff $U_1(s, q) = t$; $(s, q) \in E_3$ iff $U_1(s, q) \leq t$

Given the above-mentioned graphs which are built based on the game g and the efficiency parameter t, we can show the connection between stable social laws and standard graph-theoretic concepts:

Theorem 6.3: *Given a game g and an efficiency parameter t, the corresponding graphs G_1, G_2, G_3 satisfy the following: a stable social law for g exists iff there*

is a subset V' of the nodes of the related graphs, such that V' is a clique in G_1, a dominating set in G_3, and all nodes in V' are non-isolated, relative to V', in G_2.

The above theorem supplies an additional graph-theoretic understanding of the notion of stable social laws. A further look at such representations enables us to prove additional general existence theorems for stable social laws. One interesting general type of multi-agent encounters refers to games which are a combination of pure coordination and zero-sum games. The importance of such type of games is obvious; they allow agents either to agree and obtain "reasonable payoff" or to "fight" for "high payoff" taking the risk of obtaining "low payoff". These basic games are formally defined as follows:

Definition 6.4: Assuming w.l.o.g that the efficiency parameter t equals 0, a symmetric game g is a *mixed coordination-competition game*, if the utility functions satisfy:

1. $U_1(s,s) = 0$ for every $s \in S$.
2. $U_1(s,q) > 0$ iff $U_1(q,s) < 0$ for every $s,q \in S$.

An interesting point about mixed coordination-competition games is that they can be represented by a single graph, $\bar{G}$, which is defined as follows: the nodes of $\bar{G} = (\bar{V}, \bar{E})$ corresponds to the different strategies, and the set of edges $\bar{E}$ is defined as follows: $(s,t) \in \bar{E}$ iff $U_1(s,t) < 0$. Given this graph structure, we can prove the existence of stable social laws for an interesting class of encounters:

Theorem 6.5 : *Given a mixed coordination-competition game g, if the corresponding graph $\bar{G}$ has a sink or is 2-colorable then an appropriate stable social law exists.*

7 Discussion

In this work we have introduced a theory of stable social laws, or qualitative equilibria, for risk-averse agents. Our work bridges some of the gap between work on Artificial Social Systems and work on conflict resolution in Game Theory and AI. Social laws have been shown to be a basic and useful tool for the coordination of multi-agent systems (Moses & Tennenholtz 1990; Shoham & Tennenholtz 1995; Briggs & Cook 1995; Minsky 1991). However, the stability of social laws in a system of rational agents has been neglected so far. This work extends previous work on social laws for artificial agent societies by considering stable social laws for multi-agent encounters.

Two major lines of research related to our work are work in the field of Game Theory and work in the field of Distributed Artificial Intelligence [DAI]. Related work on rational deals and negotiations in DAI (e.g., (Rosenschein & Genesereth 1985; Zlotkin & Rosenschein 1993; Kraus & Wilkenfeld 1991)) have adopted a game-theoretic perspective. A very interesting property of this work is that it considers deals among rational agents who will not deviate from agreed-upon deals.[4] In difference to this assumption, our work is concerned with agents who will deviate from agreed-upon deals if they have a rational incentive to do so.

Much work in Game Theory has been concerned with devising rational conventions for a group of rational agents; a rational agent may deviate from a prescribed joint strategy if this deviation will improve its own situation. More specifically, much work in Game Theory (Luce & Raiffa 1957; Owen 1982; Fudenberg & Tirole 1991) has been devoted to the study of equilibrium in games; an equilibrium will have the property that there is no rational incentive for an agent to deviate from the equilibrium as long as other agents stick to it. The notion of an equilibrium has been adopted to the AI and DAI literature in various settings (e.g., (Wellman 1993)), as part of a general and important attempt to introduce social and organizational metaphors into the AI context (Simon 1981; Fox 1981; Malone 1987; Durfee, Lesser, & Corkill 1987; Doyle 1983; Davis & Smith 1983; Jennings 1995; Ishida, m. Yokoo, & Gasser 1990; Cohen & Levesque 1991; Gasser 1993). A central notion in this regard is the notion of a rational agent adopted from the decision/game theory literature. Most work in Game-Theory has associated the notion of a rational agent with the notion of expected utility maximisation. This is not however the usual way a rational agent is viewed in the AI literature, such as in work on conditional planning (Warren 1976; Peot & Smith 1992; Etzioni *et al.* 1992; Genesereth & Nourbakhsh 1993; Safra & Tennenholtz 1994).

Using a Game-Theoretic terminology, in this work we developed an equilibrium theory for risk-averse agents (Luce & Raiffa 1957; Dubois & Prade 1995; Brafman & Tennenholtz 1996). It turns out that the notion of stable social laws is a powerful tool in this regard. Consider general multi-agent encounters, the notion of social law seems to serve as a useful mechanism for obtaining stable situations for risk-averse agents, similarly to the way mixed strategies serve as useful tools for expected utility maximizers. Our theory and results can therefore be interpreted both as an extension to the theory of Social Laws presented in the AI literature, as well as a contribution to the foundations of discrete/qualitative Decision/Game Theory. We hope it can lead to further cross-fertilization between these fields.

[4] This is not to say that other assumptions are not treated by the DAI literature; see for example (Sandholm & Lesser 1995).

Appendix: Sketch of Proofs

Proof of Theorem 4.1 (sketch):

The proof follows by considering the following game:

	agent 2			
agent 1	A	B	C	D
A	(1,1)	(1,1)	(2,0)	(0,2)
B	(1,1)	(1,1)	(0,2)	(2,0)
C	(2,0)	(0,2)	(0.5,0.5)	(0.75,0.25)
D	(0,2)	(2,0)	(0.25,0.75)	(0.5,0.5)

Assume the designer wishes to guarantee the payoff 1. The fact that no stable convention exists follows by case analysis. On the other hand, if we restrict both agents to perform actions taken from $\{A, B\}$ then a payoff of 1 is guaranteed for both of the agents and no deviation is rational. ∎

Proof of Theorem 4.2 (sketch):

The proof follows by considering the following game:

	agent 2	
agent 1	A	B
A	(2.5,1)	(1.5,3)
B	(2,2)	(4,0.5)

A case analysis shows that no stable social law exists in the above-mentioned game.

∎

Proof of Theorem 5.2 (sketch):

The proof that the problem is in NP is straightforward. The proof the problem is NP-hard is by resuction from 3-SAT (Garey & Johnson 1979). Given a 3-CNF formula φ we generate a game g, for which a stable social law exists if and only if φ is satisfiable. We take the efficiency parameter t to be equal to 0, and let t' be a positive real number.

With clause number i in φ we associate the strategies $c_i^1, c_i^2, \ldots, c_i^7$ and d_i; each $c_i{}^k$ is associated with a different truth assignment to clause i (there are seven such assignments), and d_i is an additional distinguished strategy which is associated with that clause. The set of strategies for each player in the game g is the union of all strategies which are associated with the different clauses in φ.

We take g to be a symmetric game, and specify the utility function of agent 1:

1. $U_1(d_i, d_j) = -t'$ for all i, j.
2. $U_1(c_i^k, c_j^l) = 0$ iff c_i^k and c_j^l correspond to consistent assignments.
3. $U_1(c_i^k, c_j^l) = -t'$ iff c_i^k and c_j^l correspond to inconsistent assignments.
4. $U_1(d_i, c_j^k) = t'$ for all $i \neq j$ and every k.
5. $U_1(c_j^k, d_i) = -t'$ for all $i \neq j$ and every k.
6. $U_1(d_i, c_i^k) = -t'$ for every i and every k.
7. $U_1(c_i^k, d_i) = t'$ for every i and every k.

Now, consider a truth assignment T which satisfies φ. We can define a social law which leaves each agent only with the strategies which their corresponding assignments are as determined by T (and with no strategy of the form d_i). It is easy to see that we get a stable social law; the social law guarantees a payoff of 0 since the agents are left only with "consistent strategies", and deviations are irrational since there is a representative strategy of the form c_i^k for each clause.

If there exists a stable social law then it can not leave the agents with strategies of the form d_i, and must leave each agent with exactly one strategy for each clause (since otherwise a deviation to some d_j would become rational, or the agents may execute "inconsistent strategies"); these strategies need to be consistent (with respect to their corresponding assignments); hence, by combining the allowed strategies (i.e., their corresponding truth assignments) into a satisfying assignment, the other direction follows as well.

∎

Proof of Theorem 5.3 (sketch):

W.l.o.g let agent 1 be the logarithmically bounded agent. We can efficiently enumerate the set of possible restrictions on its strategies since there are only polynomially many such possibilities. For each such restriction r, let us denote the set of non-prohibited strategies by $S_1(r)$. Given $S_1(r)$ we can gather the set of strategies of the other agent (i.e., agent 2) which guarantee a payoff greater than or equals to t for agent 2 and exclude from them the ones that are dominated by other strategies of that agent (2). Let us denote this set of strategies by $S_2(r)$. Now, if there are strategies in $S_1(r)$ that are better than other strategies in $S_1(r)$ or if there exists a strategy in $S_1(r)$ which does not guarantee a payoff of t (given the previously generated set of strategies for agent 2) then we should move and try a new restriction r' on the strategies of agent 1. If that's not the case then we need to check whether there is a strategy for one of the agents which is not included in $S_1(r)$ and $S_2(r)$ respectively, and may yield a better payoff for the respective agent than what is guaranteed under $S_1(r)$ and $S_2(r)$. If there is such a deviation then we should try another r' (if exists) and otherwise we should stop (an appropriate law has been found).

The above procedure exhausts in a systematic man-

ner all possible stable social laws since each possible restriction on the behavior of agent 1 is checked, and for each such restriction the most general restriction on the second agent's behavior which still may be possible is generated. Checking stability of a given set of restrictions is polynomial, and the above enumeration procedure is polynomial as well.

∎

Proof of Theorem 6.3 (sketch):

Assume that V' satisfying the above properties exists; one can easily check that by prohibiting all strategies in $V - V'$ we get a stable social law. The efficiency is guaranteed by the requirement from G_1, and the fact that no deviation is rational is guaranteed by the requirement from G_3. The fact that no allowed action can be ignored is guaranteed by the requirement from G_2.

If there exists a stable social law then a payoff greater than or equals to t should be guaranteed regardless of the (allowed) actions selected; this implies that the nodes associated with the allowed actions constitute a clique in G_1. Similarly, since no deviation is rational these nodes should correspond to a dominating set in G_3. In addition, since there is no reason to consider behaviors which are inferior to others in a stable social law we get that no node which corresponds to an allowed strategy would be isolated in G_2.

∎

Proof of Theorem 6.5 (sketch):

If there is a sink in the graph then we can choose the corresponding strategy as a convention (i.e., both agents will be required to play only the corresponding strategy). Otherwise, if the graph is 2-colorable then we can color the graph by *red* and *blue* and prohibit all (and only) red strategies (for both agents). Clearly, two blue strategies will yield the desired payoff since the graph is 2-colorable. No deviation to red strategy is rational since the graph has no sinks and neighbors of a red strategy should be blue (i.e., a deviation may result in a negative payoff).

∎

References

Bond, A. H., and Gasser, L. 1988. *Readings in Distributed Artificial Intelligence.* Ablex Publishing Corporation.

Boutilier, C. 1994. Toward a Logic for Qualitative Decision Theory. In *Proc. of the 4th International Conference on Principles of Knowledge Representation and Reasoning*, 75–86.

Brafman, R., and Tennenholtz, M. 1996. On the Foundations of Qualitative Decision Theory. In *The Proceedings of AAAI-96 (to appear).*

Briggs, W., and Cook, D. 1995. Flexible Social Laws. In *Proc. 14th International Joint Conference on Artificial Intelligence*, 688–693.

Cohen, P., and Levesque, H. 1991. Teamwork. *Nous* 25(4).

Darwiche, A., and Goldszmidt, M. 1994. On the relation between kappa calculus and probabilistic reasoning. In *Proc. 10th Conference on Uncertainty in Artificial Intelligence (UAI '94)*, 145–153.

Davis, R., and Smith, R. G. 1983. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence* 20(1):63–109.

Demazeau, Y., and Muller, J. 1990. *Decentralized AI.* North Holland/Elsevier.

Doyle, J. 1983. A Society of Mind: Multiple Perspectives, Reasoned Assumptions, and Virtual Copies. In *Proc. 8th International Joint Conference on Artificial Intelligence*, 309–314.

Dubois, D., and Prade, H. 1995. Possibility Theory as a Basis for Qualitative Decision Theory. In *Proc. 14th International Joint Conference on Artificial Intelligence*, 1924–1930.

Durfee, E. H.; Lee, J.; and Gmytrasiewicz, P. 1993. Overeager Reciprocal Rationality and Mixed Strategy Equilibiria. In *Proc. of AAAI-93*, 225–230.

Durfee, E. H.; Lesser, V. R.; and Corkill, D. D. 1987. Coherent Cooperation Among Communicating Problem Solvers. *IEEE Transactions on Computers* 36:1275–1291.

Etzioni, O.; Hanks, S.; Weld, D.; Draper, D.; Lesh, N.; and Williamson, M. 1992. An Approach to Planning with Incomplete Information. In *Proceedings of the 3rd Conference on Principles of Knowledge Representation and Reasoning*, 115–125.

Fox, M. S. 1981. An organizational view of distributed systems. *IEEE Trans. Sys., Man., Cyber.* 11:70–80.

Fudenberg, D., and Tirole, J. 1991. *Game Theory.* MIT Press.

Garey, M., and Johnson, D. 1979. *Computers and Intractability - A Guide to the Theory of NP-completeness.* W.H. Freeman and Company.

Gasser, L. 1993. Social knowledge and social action: Heterogeneity in practice. In *Proc. 13th International Joint Conference on Artificial Intelligence*, 751–757.

Genesereth, M., and Nourbakhsh, I. R. 1993. Time Saving Tips for Problem Solving with Incomplete Information. In *Proc. of AAAI-93.*

Greenberg, J. 1990. *The Theory of Social Situations; An Alternative Game-Theoretic Approach.* Cambridge University Press.

Ishida, T.; m. Yokoo; and Gasser, L. 1990. An Organizational Approach to Adaptive Production Systems. In *Proc. of AAAI-90*, 52–58.

Jennings, N. 1995. Controlling Coopeartive Problem Solving in Industrial Multi-Agent Systems Using Joint Intentions. *Artificial Intelligence* 74(2).

Kraus, S., and Wilkenfeld, J. 1991. The Function of Time in Cooperative Negotiations. In *Proc. of AAAI-91*, 179–184.

Luce, R. D., and Raiffa, H. 1957. *Games and Decisions- Introduction and Critical Survey.* John Wiley and Sons.

Malone, T. W. 1987. Modeling Coordination in Organizations and Markets. *Management Science* 33(10):1317–1332.

Minsky, N. 1991. The imposition of protocols over open distributed systems. *IEEE Transactions on Software Engineering* 17(2):183–195.

Moses, Y., and Tennenholtz, M. 1990. Artificial Social Systems Part I: Basic Principles. Technical Report CS90-12, Weizmann Institute.

Moses, Y., and Tennenholtz, M. 1995. Artificial Social Systems. *Computers and Artificial Intelligence* 14(6):533–562.

Owen, G. 1982. *Game Theory (2nd Ed.).* Academic Press.

Peot, M. A., and Smith, D. 1992. Conditional Nonlinear Planning. In *Proceedings of the 1st International Conference on AI Planning Systems*, 189–197.

Rosenschein, J. S., and Genesereth, M. R. 1985. Deals Among Rational Agents. In *Proc. 9th International Joint Conference on Artificial Intelligence*, 91–99.

Safra, S., and Tennenholtz, M. 1994. On Planning while Learning. *Journal of Artificial Intelligence Research* 2:111–129.

Sandholm, T., and Lesser, V. 1995. Equilibrium Analysis of the Possibilities of Unenforced Exchange in Multiagent Syustems. In *Proc. 14th International Joint Conference on Artificial Intelligence*, 694–701.

Shoham, Y., and Tennenholtz, M. 1992. On Traffic Laws for Mobile Robots. Proc. of the 1st Conference on AI planning systems (AIPS-92).

Shoham, Y., and Tennenholtz, M. 1995. Social Laws for Artificial Agent Societies: Off-line Design. *Artificial Intelligence* 73.

Simon, H. A. 1981. *The Sciences of the Artificial.* The MIT Press.

Tan, S., and Pearl, J. 1994. Specification and Evaluation of Preferences under Uncertainty. In *Proc. of the 4th International Conference on Principles of Knowledge Representation and Reasoning*, 530–539.

Warren, D. H. D. 1976. Generating Conditional Plans and Programs. In *Proceedings of the Summer Conference on AI and Simulation of Behavior, Edinburgh.*

Wellman, M. P. 1993. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research* 1:1–23.

Zlotkin, G., and Rosenschein, J. S. 1993. A Domain Theory for Task Oriented Negotiation. In *Proc. 13th International Joint Conference on Artificial Intelligence*, 416–422.

Multiple Perspective Reasoning

Tze-Yun Leong
Medical Computing Laboratory
Department of Information Systems and Computer Science
National University of Singapore
Lower Kent Ridge Road
Singapore 119260
leongty@iscs.nus.sg

Abstract

Reasoning often involves deliberations in different perspectives. Distinct perspectives or views support knowledge acquisition and representation suitable for different types of inference in the same discourse. In decision analysis, for instance, state transition diagrams, influence diagrams, and decision trees provide alternative perspectives to the same decision problem; the different frameworks are best suited for different stages of decision modeling, evaluation, and analysis. This paper presents a new methodology for decision making over time and under uncertainty. We identify a formal basis that supports representation and reasoning from different perspectives. We address how to integrate information from various prospects. We propose how to incrementally extend the reasoning ontology. We also discuss the practical potential of the proposed paradigm.

1 INTRODUCTION

Dynamic decision making concerns problems in which time and uncertainty are explicitly considered. For example, a common medical decision is to choose an optimal course of treatment for a patient whose conditions may vary over time; a common financial investment decision is to determine an optimal portfolio with respect to fluctuating market factors over time.

Research in control theory, operations research, decision analysis, artificial intelligence (AI), and other disciplines has led to various techniques for formulating, solving, and analyzing dynamic decision problems. These techniques include Markov decision processes (MDPs), dynamic decision models such as dynamic influence diagrams (Tatman and Shachter 1990), Markov cycle trees (Beck and Pauker 1983), and stochastic trees (Hazen 1992), and planning in AI. They adopt different assumptions, support different ontologies, and have different strengths and weaknesses.

Reasoning about dynamic decision problems often involves integrating information from different perspectives or viewpoints. For instance, at one stage of problem deliberation, it might be essential to consider the possible physiological states that a patient would go through; at another stage, it would be illuminating to estimate the uncertain effects of a treatment that would lead to different physiological states.

One major difficulty of decision modeling is to fit the complex decision factors into simple, parameterized models. Many assumptions and constraints are implicit in the resulting models. The limited decision vocabulary contributes toward solution efficiency, but obscures model transparency. Moreover, all of the existing frameworks support single perspective reasoning; the decision models conform to the fixed vocabulary or graphical presentation convention of the chosen framework. For instance, states of the world are represented as state nodes in a Markov state transition diagram, as combinations of outcome values of the chance nodes in a dynamic influence diagram, or as branches of the Markov nodes in a Markov cycle tree.

We introduce a general dynamic decision modeling paradigm that supports multiple perspective reasoning and incremental language extension. Multiple perspective reasoning allows the modeler to visualize and examine the same information in different ways; it facilitates effective formulation and analysis of dynamic decision problems. Incremental language extension enables the framework to be customized through the use of *translators*; the scope of the dynamic decision problems addressed can therefore be gradually expanded.

This paper illustrates a language design, called DynaMoL (Dynamic decision Modeling Language), and a prototype architecture that adopt the proposed paradigm. We demonstrate the practical promise of the system via a brief description of a comprehensive case study in medicine. We address how to integrate and organize information from

different perspectives. We also examine how to extend the framework and discuss some future research issues.

2 DESIGN MOTIVATION

The DynaMoL design is motivated by an in-depth analysis of existing frameworks (Leong 1993). An effective language for dynamic decision making should provide representational and inferential support for the different tasks involved, while balancing the trade-off between model transparency and solution efficiency. In particular, such a language should satisfy the following desiderata:

First, the decision ontology must address a wide range of issues in typical dynamic decision problems. The language components and their organization should be expressive, succinct, and transparent.

Second, the language should have a formal theoretical basis with simple and rigorous syntax and semantics. These qualities would facilitate examining and analyzing the formal properties of the problem and its solutions.

Third, the language must support reasoning at multiple levels of abstraction, so that a user can deal mainly with the relevant ontological concepts, instead of the specific mathematical definitions.

Fourth, the language must support graphical visualization of the decision factors and constraints in multiple perspectives and across different levels of abstraction. Visualization in multiple perspectives reflects a common pattern in human decision making.

Fifth, the language should be incrementally extensible. Extensions involve incorporating additional language constructs for new types of decision factors and constraints. Modularity should be preserved through the translation across multiple levels of abstraction and multiple perspectives of visualization.

Sixth, the language should be systematically adaptable. Adaptation involves changes in the organization of the language constructs; it does not necessarily affect the expressiveness of the language.

Finally, the language should support implementations that can be put into practical use. A practical language is modular, comprehensive, and explicit.

3 DESIGN APPROACH

To address the transparency-efficiency trade-off, the DynaMoL design aims to separate the representation and inference support for modeling and solving dynamic decision problems. It incorporates the general idea of programming language design: translating a high level language for modeling into an object language for execution.

The language has four major components: a *dynamic decision grammar*, a *graphical presentation convention*, a *formal mathematical representation*, and a *translation convention*. The decision grammar supports problem formulation with multiple interfaces. The presentation convention, in the tradition of graphical decision models, governs parameter visualization and specification in multiple perspectives. The mathematical representation, in terms of a semi-Markov decision process (SMDP), provides a concise formulation of the decision problem; it also admits various solution methods. The translation convention establishes correspondence among the different perspectives and representations of the decision factors and constraints. Figure 1 shows the dynamic decision modeling paradigm adopted in the DynaMoL design.

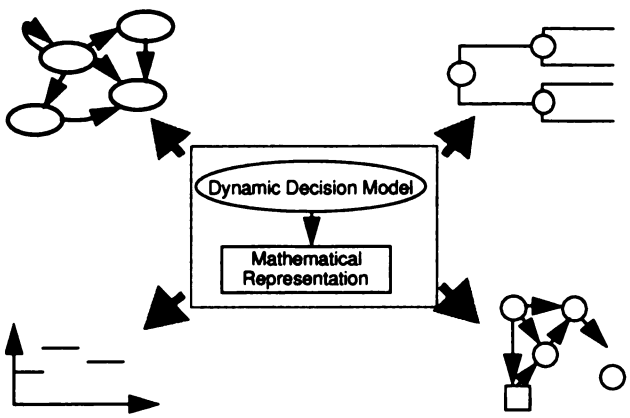


Figure 1: A new modeling paradigm.

Figure 2 shows the system architecture of a prototype implementation of DynaMoL. The system is implemented in Common Lisp with the GARNET graphics package (Myers et al. 1990). The graphical user interface allows interactive model specification. The base-language defines the model components. The translator contains the correspondence rules among the model components and the SMDP. The solver implements several solution methods for SMDPs.

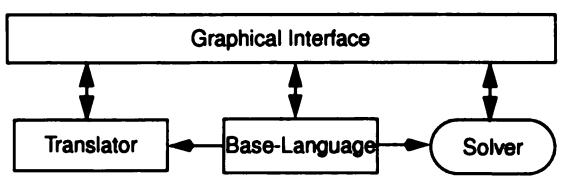


Figure 2: The prototype system architecture of DynaMoL. The blocks indicate system components; the arrows indicate information inflows.

4 DynaMoL: THE LANGUAGE

The syntax of the language is defined by the dynamic deci-

sion grammar and the graphical presentation convention. The semantics is defined by the mathematical representation of an SMDP and the translation that bridges the grammar, the presentation, and the mathematical representation of the decision factors and constraints. This section summarizes the decision ontology and the features of the language components.

4.1 BASIC DECISION ONTOLOGY

We adopt a knowledge formalization approach similar to first-order predicate calculus (FOPC), as presented in (Genesereth and Nilsson 1987). In DynaMoL, a *conceptualization* of a decision situation includes the relevant decision parameters and the interrelationships among them. Formally, a conceptualization consists of a *universe of discourse*, a *functional basis set*, and a *relational basis set*. The universe of discourse is the set of decision parameters about which knowledge is being expressed. Various functions and relations can be defined among these parameters. The functional basis set is the set of relevant functions in the universe of discourse. The relational basis set is the set of relevant relations in the universe of discourse. Based on the conceptualization, we define the expressions in DynaMoL in terms of the following components: *basic concepts*, *variable concepts*, *functions*, *relations*, and *constraints*.

4.1.1 Basic Concepts

A *basic concept* is a description of the world, within the universe of discourse, at a single point in time. There are three types of basic concepts: events, actions, and states.

Event

An *event* is a partial description of the world at a single point in time. It corresponds to a proposition or atomic sentence in FOPC. In particular, it describes an occurrence or a lack of occurrence of a phenomenon. Both "presence of red dots" and "absence of red dots" are events. In the clinical context, an event represents either an occurrence or a lack of a physical or physiological condition of one or more patients, *e.g.*, "stroke occurred" and "stroke did not occur."

Action

An *action* is a special type of events which involves an actor. The actor can be the decision maker concerned, in which the action is *controlled*; the actor can also be unknown and the action is subject to chance, in which case the action is *embedded*. In a clinical decision situation, relevant actions include different tests and treatments, and their combinations, *e.g.*, "surgery performed" and "surgery not performed." The effects of an action are expressed in terms of events, *e.g.*, a complication of the action "surgery" is the event "presence of excessive bleeding."

State

A *state* is a complete description of the world at a single point in time. The state description comprises a set of events or proposition as defined above. For instance, the state WELL in a typical clinical decision problem is defined in terms of the events: {***status*** = *alive*, ***disease-X*** = *absence*, ***disease-Y*** = *absence*, ... }. Every state has an associated *value*, indicating the desirability of being in that state.

4.1.2 Variable Concepts

A *variable concept* corresponds to a random variable in probability theory. It represents the uncertainty on centain aspects of the world, as described by the basic concepts, at a single time point. We only consider discrete variables in this work.

Chance Variable

A *chance variable* corresponds to a chance node in a decision model. It has several possible outcomes or values; the outcomes can be events or states. Each possible outcome occurs only by chance and cannot be explicitly chosen. There are three types of chance variables: event variables, state attribute variables, and state variables.

a) Event Variable

The outcomes of an *event variable* are events. An event variable usually represents a possible consequence of an action. In particular, it can capture an observable or unobservable phenomenon that directly or indirectly describes the states.

b) State Attribute Variable

A *state attribute variable* is a special kind of event variables; it represents a characteristic or property directly relevant to describing the states. The outcomes of a state attribute variable are events. Given a set of such variables, a state is usually defined in terms of the Cartesian product of their outcomes. Some heuristics, however, may be used to reduce the number of the resulting states. For example, in a typical clinical decision problem, any outcome combinations with the outcome "dead" for the state attribute variable ***status*** will result in the DEAD state.

c) State Variable

A *state variable* represents the uncertainty about the actual state that the world is in at a single point in time. It corresponds to a *state space* in an SMDP and a state variable node in a dynamic influence diagram at that time point. All the outcomes of a state variable are states.

Action Variable

An *action variable* denotes a decision or choice point at a single point in time; it corresponds to part or all of the *ac-*

tion space in an SMDP or a decision node in a dynamic influence diagram at that time point or decision stage. An action variable has several possible *alternatives*, which are actions. Any alternative action can be explicitly controlled or chosen as the decision.

Function Variable

A *function variable* denotes the value function or desirability measure structure at a single time point. It corresponds to a value node in a dynamic influence diagram at that time point or decision stage. Each outcome of the function variable is a number, a table, or a function representing the value or desirability of a state given an action. In other words, the outcomes of the variable are determined by functional combination of the outcomes of a state variable and those of an action variable.

4.1.3 Functions

The basic *functions* defined over the concepts and the relations are probability functions and value functions.

Probability Function

A *probability function* is either a probability mass function (PMF) or a cumulative distribution function (CDF).

Value Function

A *value function* measures the desirability of a state. It may have different dimensions, *e.g.*, monetary cost and life expectancy, and is usually conditional on an action.

4.1.4 Relations

The basic *relations* among the concepts are probabilistic dependences and temporal dependences.

Probabilistic Dependences

A *probabilistic dependence* relation between two concepts corresponds to the conditional dependence notion in probability theory. Such a relation indicates that a concept is conditionally dependent on another. The direction of a probabilistic dependence relation reflects the definition of the underlying conditional dependence; it has no temporal implication.

Temporal Dependences

A *temporal dependence* relation between two concepts indicates that one temporally precedes another; it has no causal implication.

4.1.5 Constraints

The *constraints* are meta-level descriptive or prescriptive conditions imposed on the concepts, the relations, and the functions as defined above. They correspond to the logical sentences and quantification sentences in FOPC. Some examples are the applicability of actions with respect to the states or time or both, and the validity of states and events with respect to each other or time or both.

4.2 DYNAMIC DECISION GRAMMAR

The complete dynamic decision grammar for DynaMoL is documented in (Leong 1994); it contains the following components:

- A finite set of names of *constructs*;
- A finite set of *productions*, each associated with a construct.

The structure of a construct can be specified in terms of "aggregate" productions, *i.e.*, the constructs have specimens comprising a fixed number of components, "choice" productions, "list" productions, etc.

The DynaMoL grammar is an *abstract grammar*; it allows the decision ontology to be extended incrementally and implemented in multiple ways. The grammar defines the structure of a dynamic decision model in terms of its components; the structures of these components are recursively defined in a similar manner. There can be many ways to manipulate such information. In other words, the abstract grammar can support different interface implementations. Moreover, the grammar can be extended by defining appropriate translators for the new constructs.

4.3 GRAPHICAL PRESENTATION

The presentation convention in DynaMoL prescribes how the decision factors and constraints in the grammar are displayed; it determines how the same information can be visualized in multiple perspectives. The current convention includes two established graphical representations. Additional perspectives such as partial decision tree view, information flow view, and two-dimensional Cartesian plots, can also be incorporated in future.

4.3.1 Transition View

The transition view corresponds directly to the Markov state transition diagram. Given an action, the transition view depicts the possible state transitions. As shown in Figure 3, the nodes denote the states, and the arcs the possible transitions given the action.

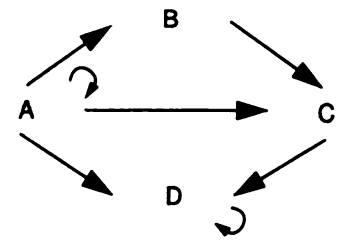


Figure 3: A transition view for an action.

4.3.2 Influence View

Given an action, the influence view depicts the possible event variables that affect the transitions from one state to another. The event variables correspond to the chance nodes in an influence diagram; the influence view, therefore, is analogous to a slice of a dynamic influence diagram, including all the chance nodes relevant to a specific decision stage. Figure 4 shows an example influence view for an action. The index "n" indicates the relevant decision stage. The nodes depict the possible event variables that affect the state transitions, and the links the probabilistic dependences. The states are captured as outcomes or values of the state variables.

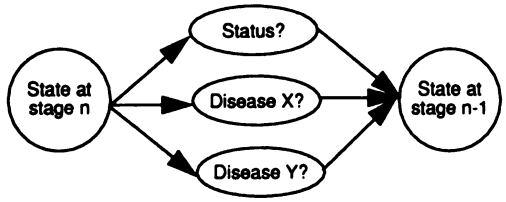


Figure 4: An influence view for an action.

4.4 MATHEMATICAL REPRESENTATION

The central idea for supporting multiple perspective reasoning is to establish a common basis among the different views or aspects of a decision situation. In DynaMoL, the theoretical basis of a dynamic decision model is an SMDP.

Briefly, an SMDP is a mathematical model of a sequential decision process. An SMDP has the following basic components: a time index set T, an action space A, a state space S, a set of one-step transition functions with PMFs $q_{ij}^{(a)}(.)$ and CDFs $Q_{ij}^{(a)}(.)$, and a set of value functions $v_i^{(a)}(.)$, where $i, j \in S, a \in A$. The stochastic nature of the transitions are reflected in both the *destination* of the transition and the *time lapsed* before the transition, both governed by the one-step transition functions. An MDP is an SMDP with constant inter-transition times at one time unit.

A solution of an SMDP is an *optimal policy;* it is a guideline for choosing the optimal actions over the decision horizon, for all possible evolutions of the states, that maximize the expected value or reward. Many solution methods are available for SMDPs.

4.5 TRANSLATION CONVENTION

Two types of translation are involved in co-ordinating multiple perspective reasoning in DynaMoL: *inter-level translation* and *inter-perspective translation.*

In inter-level translation, a model specified in the dynamic decision grammar is translated into an SMDP. The model may involve more constructs than those defining an SMDP. A set of translators or correspondence rules are employed to establish the proper correspondence. The general idea is to map a construct or a set of constructs in the grammar to an entity or relation in the mathematical representation.

Inter-perspective translation, on the other hand, establishes correspondence among the different representational formats. Since the same information is involved, inter-level translation can be defined in terms of any covering set of representation constructs among the different perspectives. Figure 5 illustrates the idea of inter-perspective translation.

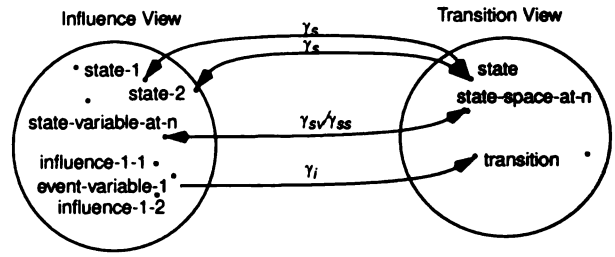


Figure 5: Translation by mapping. An arrow indicates a translator or correspondence rule.

5 A DYNAMIC DECISION MODEL

A dynamic decision model in DynaMoL is a well-formed or complete model when it corresponds to a well-formed SMDP with an *evaluation* or *optimality criterion.* Some model constructs directly correspond to the underlying mathematical definitions, others need to be translated. The default optimality criterion is finite-horizon discounted total expected value.

Definition: *(Dynamic Decision Model)* A dynamic decision model M in DynaMoL is an 8-tuple $\langle T, A, S, E, \Omega_E, \Xi, \Psi, \varsigma, K, \Gamma \rangle$ together with an *optimality criterion*, where:

- The *decision horizon* $T = \{0, 1, 2, 3, \ldots\}$ denotes the time frame for the decision problem;
- The *action space* $A = \{a_1, a_2, \ldots, a_{|A|}\}$ denotes the set of alternative actions to be considered;
- The *state space* $S = \{s_1, s_2, \ldots, s_{|S|}\}$ denotes the set of states or conditions that would affect the value functions. The states are defined in terms of a set of *state attribute variables*;
- For each action $a \in A$, its *action-specific event variable space* $E_a = \{e_1, e_2, \ldots, e_{|E_a|}\}$ denotes the set of event or chance variables that constitute its effects. The *event*

variable space $E = \bigcup_{a \in A} E_a$ denotes the set of all event variables for all the actions;

- Each event variable e has an associated *event* or *outcome space* $\Omega_e = \{\omega_1, \omega_2, \ldots, \omega_k\}; k \geq 1$, which denotes the set of possible outcomes for e. The *action-specific event space* Ω_{E_a} and the *overall event space* Ω_E are similarly defined;
- Conditional on each action, a set of possible *transitions* $\Xi = \{\xi(a, i, j, t) \mid i, j \in S,\ a \in A,\ t \in T\}$ is defined among the states;
- A transition relation $\xi \subseteq A \times S \times S \times T$ between any two states $i, j \in S$, given an action $a \in A$ at time $t \in T$, denotes the accessibility of j from i given a at t. The nature of this accessibility is characterized by a one-step transition function with PMF $q_{ij}^{(a)}(.)$ and CDF $Q_{ij}^{(a)}(.)$ as defined for an SMDP;
- Conditional on each action, a set of *probabilistic influences* $\Psi = \{\psi(a, x, y, t) \mid a \in A,\ x, y \in E_a,\ t \in T\}$ is defined among the event variables;
- A probabilistic influence relation $\psi \subseteq A \times E_a \times E_a \times T$ between any two event variables $x, y \in E_a$, given an action $a \in A$ at time $t \in T$, reflects the probabilistic dependences, conditional on the action, among the outcomes of the two event variables at time t;
- Conditional on each action, and with respect to the evaluation criterion, a set of *value functions* $\varsigma = \left\{ v_i^{(a)}(m) \middle| i \in S;\ a \in A, m = 0, 1, 2, \ldots \right\}$ are defined for the states;
- A value function $v : A \times S \times \aleph \rightarrow \Re$ determines the objective value achievable in state $i \in S$ over time duration $m \in \aleph$, conditional on action $a \in A$;
- Definitions of the above components can be subjected to a set of general or domain-specific constraints $K = \{\kappa \mid \kappa \subseteq \{S \cup \Omega_E \cup \Xi \cup \Psi \cup \varsigma\}^n; n \geq 2\}$, where κ is a constraint which may establish, among other things, the valid conditions of and the logical relationships among the actions, states, events, and their associated relations; and
- Γ is the set of *translators*, *i.e.*, the set of correspondence rules or functions that establish equivalence relations among the language constructs.

To illustrate the language components and the modeling and solution processes in the DynaMoL framework, we briefly describe a case study in medicine.

6 A CASE STUDY

Atrial fibrillation (AF) is a kind of cardiac arrhythmia or abnormal heartbeat. It has two major undesirable side-effects: hemodynamic deterioration and embolization. AF can occur in paroxysms, i.e., sudden, periodic episodes. Both the frequency and the length of the AF episodes may increase with time; constant fibrillation often develops eventually.

AF management involves using antiarrhythmic agents to control heart rates and to restore normal sinus rhythm. Because of the risk of embolization, anticoagulants such as warfarin and aspirin are often used to prevent blood clot formation in the atria. The treatments, however, have corresponding undesirable side-effects. In particular, a common antiarrhythmic agent, Quinidine, increases the risk of sudden death; an anticoagulant may cause excessive bleeding, which in turn may lead to strokes or death.

The case in question is based on an actual clinical consult at the Tufts-New England Medical Center in Boston, Massachusetts. The patient is a 52 year old white male with a history of paroxysmal AF. He has been on warfarin and on Quinidine. The clinical decision problem is as follows:

Quinidine decreases the proportion of time that the patient spends in AF. Does the decreased risk of embolic complications justify the increased risk of sudden death?

We have assessed the effectiveness of DynaMoL for modeling and solving the dynamic decision problem with respect to three criteria. First, we demonstrated how the relevant decision factors and constraints can be expressed in the DynaMoL ontology. Second, we compared the solutions to the results of the actual clinical consult or the clinical judgment of domain experts or both. Third, we conducted sensitivity analysis on the solutions to ensure reasonable behavior of the parameters involved. The detailed discussion, data interpretation, and analysis are documented in (Leong 1994).

7 MODEL FORMULATION

In DynaMoL, we formulate a dynamic decision model via interfaces that implements the dynamic decision grammar, through multiple perspective visualization.

7.1 STRUCTURE SPECIFICATION

In the case study, the decision horizon T is 600 months or 50 years. The evaluation criterion for the decision problem is life-adjusted quality expectancy (QALE).

The decision objective is to maximize the total expected QALE of the patient by choosing the best alternative among

four strategies: Start without any drug, with warfarin, with Quinidine, or with both, and stop warfarin when necessary.

The action space $A = \{\text{NoDrug, Quinidine}\}$ consists of the two externally controlled actions for the strategies involved. The effects of warfarin are modeled as embedded actions in the state descriptions.

The state space S denotes the set of states in which the controlled actions can take place. The states are defined in terms of the following state attribute variables: **Sinus rhythm** indicates heartbeat pattern. **Cerebral vascular accident** indicates history of obstruction of blood flow to the brain, which may result in a stroke or temporary weakness. **Warfarin eligibility** affects the applicability of the action warfarin. **Warfarin status** reflects the status of the embedded action warfarin.

Based on the state attribute variables, a total of 20 states are defined in the case study. For instance, a relevant state in the case study is: AF-STR-WNE, which indicates that the patient is in atrial fibrillation (AF), has a history of stroke (STR), and is not eligible for warfarin (WNE).

Conditional on each controlled action, a set of possible transitions Ξ is defined among the states. Each transition ξ is associated with a one-step transition function with PMF $q_{ij}^{(a)}(.)$ and CDF $Q_{ij}^{(a)}(.)$, governing the destination of the transition and the time remaining before the transition.

The transition view supports graphical presentation of the state transition diagram. Transition functions can be specified directly on the links representing the transitions. A simplified transition view for the action Quinidine, including only the state attributes related to the embedded action warfarin, is shown in Figure 6.

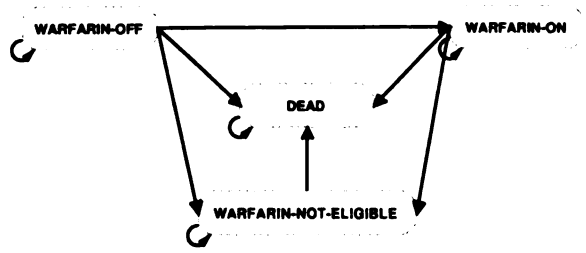


Figure 6: Transition view for the case study.

At times it is difficult to specify the transition functions directly, especially when the action effects are partially observable or unobservable. Such effects are modeled as event variables and their associated relations.

In the case study, the event variable space E includes event variables indicating the presence or absence of **thromboembolization, cerebral hemorrhage, gastro-intestinal bleeding,** and whether the patient survives the events should they occur. Conditional on an action such as Quinidine, these events probabilistically influence the state attribute variables mentioned such as **warfarin eligibility, cerebral vascular accident,** *etc.*, which in turn define the physiological states of the patient

The influence view supports graphical presentation of the event variables and probabilistic influences for an action at a particular decision stage. Conditional probabilities can be defined directly on the nodes representing the event variables. Figure 7 shows an influence view for the action Quinidine in the case study. The values or outcomes of the two state variables, represented by the nodes named "state-n" and "state," are the different states defined in the model.

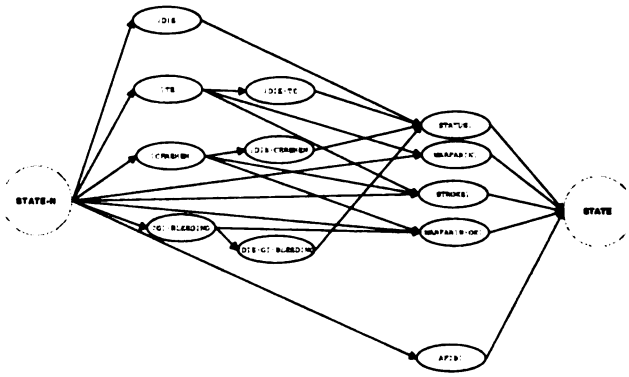


Figure 7: Influence view for action Quinidine.

Conditional on each controlled action, a set of value functions ς is defined for the states. A value function $v_i^{(a)}(.)$ indicates the QALE achievable in a state over a time duration, given the action.

7.2 CONSTRAINT DECLARATION

The components of a model in DynaMoL can be subjected to a set of general or domain-specific constraints K. The constraints impose logical relationships among the the actions, states, event variables, and their associated relations. In general, a constraint κ corresponds to a logical or quantification sentence or *well-formed formula* (wff) in FOPC.

The set of declaratory constraints concern the definitions of the states, event variables, and their associated relations. Many constraints are inherent in the definitions supported by DynaMoL. For instance, the definition of "absorbing states," or "validity of a state."

In addition, DynaMoL supports a logical language for specifying some logical relations among the language components. The constraint language is currently very limited. A major constraint incorporated is as follows:

7.2.1 The Disjunctive Definition Constraint

The *disjunctive definition*, or *"partial* OR-gate," for event combinations is analogous to a *canonical model* (Pearl, 1988) for specifying Bayesian networks. The partial OR-gate facilitates conditional distribution specifications of event variables. It allows us to express statements such as: "a patient is dead if he dies from natural causes or dies from a stroke or dies from cerebral hemorrhage or dies from gastro-intestinal bleeding." In other words, we can specify some indices of the conditional distribution table as disjunctions of events, as opposed to the usual conjunctions.

With respect to the generalized noisy-OR model developed by (Srinivas 1992), of which the binary noisy OR-gate as described in (Pearl 1988) is a special case, the partial OR-gate is different in the following ways. First, in the DynaMoL vocabulary, the former is a function on the event variable space $\boldsymbol{E}$, while the latter is a function on the event space $\Omega_{\boldsymbol{E}}$. Second, in the former, the combination constraint is imposed on an event variable and all its predecessors, while in the latter, the combination constraint can be imposed on an event variable and any subset of its predecessors, hence the name "partial."

The disjunctive definition constraint has the general form:

$$e_1 \vee e_2 \vee \ldots \vee e_k \Rightarrow e_c$$

where $e_i, e_2, \ldots, e_k$ are the *constraining events* and e_c is the *consequent event*. The constraint is read as: "If e_i or e_2 or ... or e_k, then e_c." The constraint is on the specification and interpretation of the conditional distribution table of an event variable $\boldsymbol{x}$. The constraining events are outcomes of the probabilistic predecessors of $\boldsymbol{x}$; the consequent event is an outcome of $\boldsymbol{x}$.

In DynaMoL, a user can explicitly specify the partial OR-gate constraints for the different event variables in terms of a set of logical statements as shown above. The conditional distribution tables with the correct dimensions and labels are automatically created. The numerical parameters involved can then be specified accordingly.

7.3 PARAMETRIC ASSESSMENT

The numbers, functions, and probability distributions associated with the variables and relations are usually assessed after the structure of the model is in place. Some guidelines for directly specifying one-step transition functions in an SMDP, corresponding to the transition view in DynaMoL, are documented in (Leong 1994). When direct estimation of the one-step transition functions is difficult, the parameters can be specified in the influence views, in terms of the conditional distributions of their constituent event variables instead.

For the Markov case, each entry $C[x, y]$ in a conditional distribution table or matrix C is:

$$P\{y|x, a, t\} \qquad \text{(EQ 1)}$$

subject to

$$\sum_y P\{y|x, a, t\} = 1 \qquad \text{(EQ 2)}$$

where x is a row index representing a combination of outcome events of all the probabilistic predecessors for an event variable $\boldsymbol{y}$, $y \in \Omega_{\boldsymbol{y}}$ is a column index representing an outcome of $\boldsymbol{y}$, $a \in \boldsymbol{A}$, and $t \in \boldsymbol{T}$. The entry, which is usually a function $f(t)$ of time, indicates the probability of event y in the next decision stage, given that event x occurs and action a taken at the decision stage at time t.

For the semi-Markov case, there are two ways to assess the one-step transition functions in terms of the constituent event variables. Consider specifying the PMFs of one-step transition functions, $q_{ij}^{(a)}(.)$. The PMFs of one-step transition functions can be defined either in terms of the transition probabilities $P_{ij}^{(a)}(.)$ and holding time PMFs $h_{ij}^{(a)}(.)$, such that

$$q_{ij}^{(a)}(m, t) = P_{ij}^{(a)}(t) h_{ij}^{(a)}(m, t);$$

$$a \in \boldsymbol{A}, i, j \in \boldsymbol{S}, t \in \boldsymbol{T}, m \geq 0 \qquad \text{(EQ 3)}$$

or the conditional transition probabilities $p_{ij}^{(a)}(.)$ and waiting time PMFs $w_i^{(a)}(.)$, such that

$$q_{ij}^{(a)}(m, t) = p_{ij}^{(a)}(m, t) w_i^{(a)}(m, t);$$

$$a \in \boldsymbol{A}, i, j \in \boldsymbol{S}, t \in \boldsymbol{T}, m \geq 0 \qquad \text{(EQ 4)}$$

The first approach, corresponding to (EQ 3), is to assess the probabilities of the transition destinations first, then decide on the holding times with respect to those transitions. In this case, we interpret the probability given in (EQ 1), which is a function $f(t)$ of time, as an eventual transition probability, *i.e.*, the probability of event y over those of other outcomes of $\boldsymbol{y}$, given that event x occurs and action a is taken at the decision stage at time t. The collection of conditional distributions in the influence view will then constitute the eventual transition probabilities $P_{ij}^{(a)}(t)$, where $i, j \in \boldsymbol{S}, a \in \boldsymbol{A}, t \in \boldsymbol{T}$ for the underlying SMDP. We then estimate the holding time PMFs directly for the corresponding transitions. In the clinical context, this approach first determines the transition destination of a patient, and then depending on this destination, estimates the time duration spent in the current state before making the transition. For instance, there is a probability of 0.3 that a patient who has undergone surgery A will develop complication B; the probability for such a patient to develop the complication is 0.2 in the first month after surgery, 0.5 in the second month, and 0.3 in the third month.

The second approach, corresponding to (EQ 4), is to first decide on the waiting times, and then assess the probabilities of transition destinations with respect to those waiting times. In this case, we directly estimate the waiting time PMFs for each state $s \in S$. We then interpret the probability given in (EQ 1), which is now a function $f(m, t)$ of both duration and time, as a conditional transition probability, *i.e.*, the probability of event y over those of other outcomes of y, given that event x occurs, action a is taken, and the waiting time is m at the decision stage at time t. The collection of conditional distributions in the influence view will then constitute the conditional transition probabilities $p_{ij}^{(a)}(m, t)$, where $i, j \in S, a \in A, t \in T, m \geq 0$ for the underlying SMDP. In the clinical context, this approach first estimates the time duration a patient spends in the current state before making a transition, and then depending on the this duration, determines the transition destination. For instance, the probability that a patient who has undergone surgery A will remain well is 0.2 for a month after surgery, 0.5 for two months and 0.3 for three months; if the patient is well, the probability for him to develop complication B is 0.6 in the first month after surgery, 0.5 in the second month, and 0.3 in the third month.

Validity, adequacy, and relevance of the numeric parameters are determined by established techniques when appropriate. Such techniques could be incorporated into the DynaMoL ontology in future.

8 MODEL TRANSLATION

As mentioned, a dynamic decision model is translated into an SMDP for solution and analysis. We now briefly illustrate how to employ the translators to support multiple perspective reasoning and incremental language extension in DynaMoL.

8.1 TRANSLATING TRANSITION VIEW CONSTRUCTS

When completely and consistently specified, the transition view components correspond directly to the definitions of an SMDP. Hence no special translators are needed in this case. Details of the model, however, usually cannot be easily specified directly in the transition view; inter-perspective translations are almost always involved in translating constructs in other views into those of the transition view.

8.2 TRANSLATING EVENT VARIABLES AND INFLUENCES

The mathematical definitions of SMDP do not include event variables and influences. The probabilities involved in the event variables and the corresponding influences between any two states, therefore, should be translated into the one-step transition functions characterizing the transitions between the two states.

The *influence view translator* handles this task. The algorithm is based on the expectation of conditional probabilities; it is analogous to the chance node reduction algorithm in influence diagrams (Shachter 1986).

Given random variables x, y, and z, conditional expectation of z given x with respect to y means, for $x \in \Omega_x$, $y \in \Omega_y$, $z \in \Omega_z$, where Ω_e denotes the set of outcomes or values for an event variable e:

$$P\{z|x\} = \sum_i P\{z|y_i\} \cdot P\{y_i|x\} \qquad \text{(EQ 5)}$$

With reference to Figure 7 and Figure 8, the overall idea of the algorithm is to reduce the intermediate event variable nodes between the two state variable nodes, so that the final diagram contains only a direct influence between the two state variables. Assuming static action space and static state space, the conditional distribution table associated with the state variable on the right contains the set of PMFs or CDFs for the one-step transition functions or its components, conditional on an action.

Figure 8: Final influence view after reduction.

In essence, the influence view translator iteratively identifies an event variable node to be reduced, updates the conditional distributions of other event variables affected by it, and removes it. The main algorithm is as follows:

INFLUENCE-VIEW-TRANSLATOR (ID)

```
◊ ID is a sequence of event-variable-nodes
x ← FIND-REMOVABLE-EVENT-VARIABLE (ID)
  while x
    do
   ID ← ABSORB-EVENT-VARIABLE (ID, x)
    x ← FIND-REMOVABLE-EVENT-VARIABLE
(ID)
  return ID
```

8.3 MAINTAINING CONSISTENCY

Decision modeling in DynaMoL involves working with different graphical views at the same time. While the views should reflect the same information, modifying the constructs in one view may render the information in another view obsolete. This calls for some measures of

consistency checking and management. There are two types of consistency checking in DynaMoL:

Intra-view consistency ensures that the structural and numerical parameters in a particular view are specified correctly, *e.g.*, the probabilities add up to 1, the number of states displayed are in accordance with the state-space, *etc.* Such consistency is maintained by incorporating simple checking mechanisms on the interfaces.

Inter-view consistency ensures that updated information is reflected in all the relevant views when the translators are invoked. Most of the translators are unidirectional, *e.g.*, translating event variables and influences into transition functions. When such a translator is invoked, we have to make sure that the target view is updated, *e.g.*, new transition arcs with associated one-step transition functions are displayed in the transition view. Under this convention, however, information of the translation origin may become obsolete when the target is modified, *e.g.*, updating a transition function directly after translating the influence view. A consistency table can keep track of the translations invoked during a modeling session, and a flag is signalled if the relevant target information is modified.

For bi-directional translations, *e.g.*, updating parameters with direct correspondence in different views, modifications are reflected throughout the model; inconvenience is minimized by requiring that changes are reflected only when they are saved or confirmed.

9 MODEL SOLUTION AND ANALYSIS

Problem solution in DynaMoL involves solving the corresponding SMDP. A variety of solution methods are available for SMDPs. Applicability of the different solution methods depends on the problem characteristics. The solution methods implemented in the prototype system are based on value iteration of the dynamic programming equation (Howard 1971). (EQ 6) shows the general form of the optimality equation for discrete-time MDPs, with an arbitrary discount factor $0 < \beta \leq 1$:

$$V_i^*(\beta, t) = max_a \left\{ \beta^t v_i^{(a)}(1) + \sum_j P_{ij}^{(a)}(t) \cdot V_j^*(\beta, t+1) \right\}; \quad i, j \in S, a \in A, t \in T \qquad \text{(EQ 6)}$$

where $P_{ij}^{(a)}(t)$ is the transition probability from state i to state j, given action a, at time t. The solution to (EQ 6) is an optimal policy that maximizes $V_{init}^*(0)$, the optimal expected value or reward for an initial state at time 0.

Default solution methods in existing dynamic decision modeling techniques are all based on value iteration. These methods, however, are usually restricted by the graphical structure of the corresponding models. By dealing directly with the mathematical definitions of SMDPs, the DynaMoL design potentially admits all existing and future solution methods for SMDPs.

For the case study, the solution produced by the solver is a set of two policies, corresponding to the four strategies considered; each policy includes the expected value achievable for all possible starting states and all possible decision stages. We assume that the patient has a probability of 0.25 to be in atrial fibrillation, has no history of thromboembolism, and is not on warfarin at the beginning of the decision horizon. The solution indicates that the strategy that administers only warfarin initially is the preferred strategy over a long-term decision horizon of 50 years or 600 months. Quinidine does not decrease the risk of embolic complications enough to justify the increased risk of sudden death.

Structural and parametric analysis can be done by changing the event variables. The influence view supports such local structural changes. In addition to determining and analyzing the information involved in the changes, the revised models will also be translated into and solved as SMDPs to examine the changes in solution results. On the other hand, the transition view facilitates detection of improperly specified parameters. For instance, there should not be a transition from a state where warfarin is ineligible to one where warfarin is administered.

Only numerical parametric analysis is conducted in the case study. This is done by simply changing some of the numbers and invoking the solution method again.The results indicate that the strategy of administering only warfarin to the patient is the dominant strategy for all reasonable ranges of the numerical parameters involved.

As compared to the original solutions for the clinical consult, our prototype system achieves correct answers and provides extra capabilities in terms of expressiveness and efficiency for modeling and solving the decision problem. For instance, DynaMoL allows direct accounting of varying relative risk of bleeding for warfarin with respect to the duration of treatment; expression of such duration-dependent factors are difficult in most existing decision modeling frameworks.

10 RELATED WORK

(Egar and Musen 1993) has examined the grammar or higher level language approach to specifying decision model variables and constraints. This grammar is based on the graphical structure of influence diagrams; it captures prototypical patterns at a high level abstraction for modeling trade-offs or dilemmas in clinical decision problems. The grammar, however, has not been extended to handle dynamic decision models.

Recent efforts in decision-theoretic planning have devised methods based on MDPs for planning in stochastic do-

mains. These works focus on developing and bounding approximate solution methods (Dean et al. 1993a), (Dean et al. 1993b), (Haddawy et al. 1995). They apply the mathematical formulation directly, and do not address the general dynamic decision ontology.

Graphical representation of continuous-time semi-Markov processes has been explored in (Berzuini et al. 1989) and (Dean et al. 1992). Similarly, recent work on dynamic network models (Dagum et al. 1992), (Dagum and Galper 1993) capture the general time-series analysis techniques in Bayesian networks. These frameworks, however, focus only on single-perspective presentation of the relevant decision and probabilistic variables as Bayesian networks or influence diagrams.

On integrating dynamic decision model and SMDPs, (Provan and Clarke 1993), (Provan and Poole 1991) have developed techniques that automatically construct dynamic influence diagrams from the data of underlying SMDPs, but they only employ algorithms for evaluating dynamic influence diagrams. In this framework, the state attribute variables involved in each decision stage are represented as a Bayesian network; the overall structure corresponds to the dynamic influence diagram representation of an SMDP. Tailoring of parameters, however, has to be explicitly done for each decision stage.

11 DISCUSSION

We presented a new dynamic decision modeling paradigm that supports multiple perspective reasoning. The practical promise of the DynaMoL framework has been demonstrated through a prototype implementation and a comprehensive case study in medicine.

DynaMoL is more expressive and efficient than existing dynamic decision modeling frameworks. The high-level modeling constructs and the multiple graphical perspectives allow simple and explicit specification of the decision factors and constraints; the underlying SMDP is a more general mathematical model than the MDP that underlies most existing frameworks. The prototype system can handling a large class of dynamic decision problems in medicine and other domains. The system is being used to model various real-life decision problems in clinical and pharmaceutical decision analysis, *e.g.*, follow-up studies of colorectal cancer, diabetes management, evaluation of new drugs for AIDS, *etc.* Applications in other domains are also being planned.

Our immediate technical challenges fall into three categories. First, we will enhance the expressiveness of the modeling language to handle more complex decision situations; this includes extending the decision vocabulary and adding more visualization perspectives, *e.g.*, a *tree view* which allows expression of asymmetric probabilistic relationships. Second, we will improve the efficiency of the solution methods; this includes analyzing model and solution complexities under different decision situations, incorporating more solution methods, and developing new solution methods for time-critical decision situations. Third, we will provide automated support for many aspects of the modeling and solution processes; this includes interactively or automatically derive decision parameters from distributed knowledge sources, *e.g.*, large medical databases, medical information depositories on the Internet, and human experts.

Our immediate practical challenges include identifying the types of decision problems that the proposed framework can or cannot adequately handle, building a sizeable user group, and implementing a stable system for routine use.

Acknowledgments

I would like to thank Drs. Charles E. Ellis and Stephen G. Pauker for guidance in conducting the case study.

This paper reports work done mainly at the Clinical Decision Making Group, MIT Laboratory for Computer Science, Cambridge, MA, USA. This research was supported by the National Institutes of Health Grant No. R01 LM04493 from the National Library of Medicine, and by the USAF Rome Laboratory and DARPA under the contract F30602-91-C-0018. It was also supported by the National University of Singapore under an Academic Research Grant No. RP940645.

References

Beck, J. R. and Pauker, S. G. (1983). The Markov process in medical prognosis. *Med. Decision Making*, 3:419–458.

Berzuini, C., Bellazzi, R., and Quaglini, S. (1989). Temporal reasoning with probabilities. In *Proceedings of the Fifth Workshop on Uncertainty in Artificial Intelligence*, pages 14–21. Assoc. for Uncertainty in Artificial Intelligence.

Dagum, P. and Galper, A. (1993). Forecasting sleep apnea with dynamic network models. In Heckerman, D. and Mamdami, A., editors, *Uncertainty in Artificial Intelligence: Proceedings of the Ninth Conference*, pages 64–71, San Mateo, CA. Morgan Kaufmann.

Dagum, P., Galper, A., and Horvitz, E. (1992). Dynamic network models for forecasting. In Dubois, D., Wellman, M. P., D'Ambrosio, B. D., and Smets, P., eds, *Uncertainty in Artificial Intelligence: Proceedings of the Eighth Conference*, pages 41–48, San Mateo, CA. Morgan Kaufmann.

Dean, T., Kaelbling, L. P., Kirman, J., and Nicholson, A. (1993a). Deliberation scheduling for time-critical sequential decision making. In Heckerman, D. and Mamdami, A., eds, *Uncertainty in Artificial Intelligence: Proceedings of the Ninth Conference*, pages 309–316, San Mateo, CA. Morgan Kaufmann.

Dean, T., Kaelbling, L. P., Kirman, J., and Nicholson, A. (1993b). Planning with deadlines in stochastic domains. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 574–579.

Dean, T., Kirman, J., and Kanazawa, K. (1992). Probabilistic network representations of continuous-time stochastic processes for applications in planning and control. In *Proceedings of the First International Conference on AI Planning Systems*.

Egar, J. W. and Musen, M. A. (1993). Graph-grammar assistance for automated generation of influence diagrams. In Heckerman, D. and Mamdami, A., editors, *Uncertainty in Artificial Intelligence: Proceedings of the Ninth Conference*, pages 235–242, San Mateo, CA. Morgan Kaufmann.

Genesereth, M. R. and Nilsson, N. J. (1987). *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, San Mateo, CA.

Haddawy, P., Doan, A., and Goodwin, R. (1995). Efficient decision-theoretic planning: Techniques and empirical analysis. In Besnard, P. and Hanks, S., editors, *Uncertainty in Artificial Intelligence: Proceedings of the Eleventh Conference*, pages 229–236.

Hazen, G. B. (1992). Stochastic trees: A new technique for temporal medical decision modeling. *Medical Decision Making*, 12:163–178.

Howard, R. A. (1971). *Dynamic Probabilistic Systems*, volume 1 & 2. John Wiley and Sons, New York.

Leong, T.-Y. (1993). Dynamic decision modeling in medicine: A critique of existing formalisms. In *Proceedings of the Seventeenth Annual Symposium on Computer Applications in Medical Care*, pages 478–484. IEEE.

Leong, T.-Y. (1994). An integrated approach to dynamic decision making under uncertainty. Technical Report TR 631, MIT Laboratory for Computer Science, 545 Technology Square, Cambridge, MA 02139.

Myers, B. A., Giuse, D. A., Dannenberg, B. V. Z., Kosbie, D. S., Pervin, E., Mickish, A., and Marchal, P. (1990). Garnet: Comprehensive support for graphical, highly-interactive user interfaces. *IEEE Computer*, 23(11):71–85.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA.

Provan, G. M. and Clarke, J. R. (1993). Dynamic network construction and updating techniques for the diagnosis of acute abdominal pain. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(3):299–307.

Provan, G. M. and Poole, D. (1991). A utility-based analysis of consistency-based diagnosis. In Allen, J., Fikes, R., and Sandewall, E., editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference (KR'91)*, pages 461–472, San Mateo, CA. Morgan Kaufmann.

Shachter, R. D. (1986). Evaluating influence diagrams. *Operations Research*, 34:871–882.

Srinivas, S. (1992). Generalizing the noisy or model to n-ary variables. Technical Memo 79, Rockwell International Science Center, Palo Alto Lab., Palo Alto, CA.

Tatman, J. A. and Shachter, R. D. (1990). Dynamic programming and influence diagrams. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(2):365–379.

Reports on Implementations

Parallel Transitive Reasoning in Mixed Relational Hierarchies

Eunice (Yugyung) Lee and James Geller
Department of Computer and Information Sciences
New Jersey Institute of Technology
Newark, NJ 07102

Abstract

Class hierarchies have been used traditionally in Knowledge Representation and Reasoning for a number of purposes such as inheritance, classification and transitive closure reasoning. In the last several years we have been following two lines of investigation to extend this kind of research. From a conceptual level we have worked on techniques to extend such reasoning to hierarchies other than the IS-A hierarchy. Specifically, we have developed a model of inheritance for part hierarchies (Halper 1992, Halper 1993, Halper 1994). From an implementation point of view we have worked on building fast reasoners based on massively parallel representations of IS-A, Part-of, Contained-in, etc. hierarchies (Lee 1993, Lee 1995, Lee 1996a, Geller 1991a, Geller 1991b, Geller 1993a, Geller 1993b, Geller 1994a, Geller 1994b). However, in all this work we assumed that there exist separate hierarchies. In an often-cited paper by Winston, Chaffin, and Hermann (Winston 1987) a model of reasoning is introduced that permits the combination of IS-A, Part-of, and Contained-in in a single hierarchy. The purpose of our paper is to present representational constructs and reasoning algorithms that combine these three ingredients: mixed relation hierarchies, transitive closure reasoning, and massively parallel algorithms. It is hoped that this combination will lead to progress both in better approximating human commonsense reasoning and in better approximating human speed of reasoning. We conclude the paper with a brief description of a medical vocabulary that we have been using as a source of test data.

1 INTRODUCTION

The use of IS-A hierarchies has been a main staple in Knowledge Representation and Reasoning since Quillian's inception of semantic networks (Quillian 1968). The number of papers that deals with other hierarchies of transitive binary relations is, comparatively, much smaller. However, the Part-of hierarchy has received some attention (Schubert 1979, Schubert 1983, Schubert 1987). Specifically in (Schubert 1987) the importance of transitive closure reasoning in IS-A and Part-of hierarchies has been well explained. We have previously done research on reasoning in Part-of hierarchies, especially inheritance reasoning (Halper 1992, Halper 1993, Halper 1994).

Schubert *et al.*'s approach to transitive closure reasoning has been an inspiration for us for another reason: It permits transitive closure reasoning in constant time, practically independent of the size of the knowledge base. This scalability of reasoning power with growing knowledge bases has been recognized as a very important factor for improving implemented Artificial Intelligence (Hendler 1995). In the past several years we have developed techniques and algorithms for dealing with two weaknesses of Schubert's approach: (1) Update algorithms are not independent of the size of the knowledge base, and (2) Schubert's original approach is limited to trees. We have dealt with problem (1) by using massive parallelism, following the paradigms of Shastri, Kitano, Waltz, Hendler, and Evett (Shastri 1988, Shastri 1989, Shastri 1990, Kitano 1991, Kitano 1991b, Kitano 1991d, Kitano 1991e, Waltz 1990, Evett 1989, Evett 1993a, Evett 1993b). We have dealt with problem (2) by adapting a representation of Agrawal *et al.* (Agrawal 1989, Agrawal 1990) that permits directed acyclic graphs instead of trees. As a result we have developed and implemented massively parallel algorithms for fast retrieval and update in hierarchies of any kind of binary transitive relation (Lee 1993, Lee

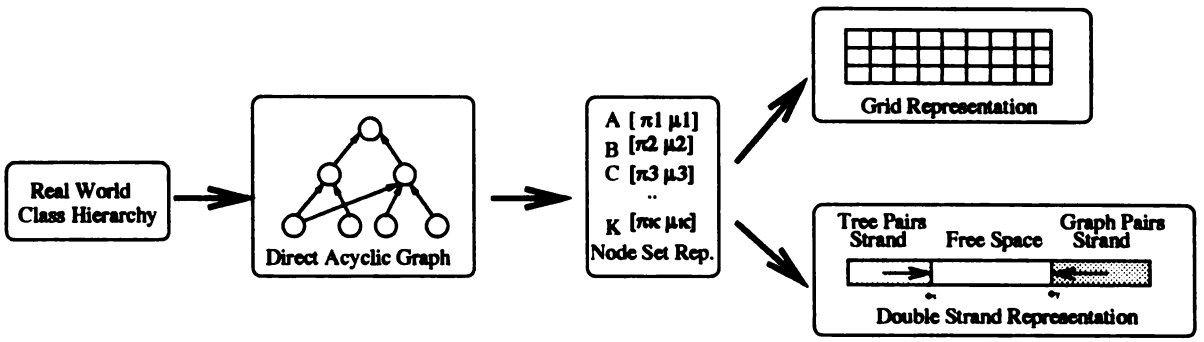


Figure 1: Three Step Mapping

1995, Lee 1996a, Geller 1991a, Geller 1991b, Geller 1993a, Geller 1993b, Geller 1994b, Geller 1994a).

In this paper we are adding another ingredient to our research. In (Winston 1987) it was pointed out that it makes sense to combine different binary transitive relations into a single reasoning process. However, not every conclusion that can be drawn in such a case is correct. Winston *et al.* describe a condition when such reasoning is correct.

As an example, if the following premises are given:

(1) Wings are parts of birds.
(2) Birds are creatures.

We can consider the following two conclusions:

(3) Wings are parts of creature.
(4) Wings are creatures.

We have obtained a reasonable conclusion (3) while (4) is an invalid conclusion (Winston 1987).

Winston *et al.* introduced a hierarchical priority ordering among the hierarchical relations (Winston 1987), such that mixed inclusion relation syllogisms are valid if and only if the conclusion expresses the lower relational priority appearing in the premises. For an alternative approach to transitivity reasoning with mixed relations, see (Cohen 1988).

We have encountered a practical example of this kind of reasoning in our test bed domain which will be described now. J. Cimino *et al.* of Columbia Presbyterian Medical Center have created a large Medical Entities Dictionary (MED) on top of a semantic network model (Cimino 1989, Cimino 1992, Cimino 1993). The MED is a very large vocabulary, incorporating over 43,000 terms, 55,000 IS-A links, 2,500 Part-of links, etc. We have chosen to focus on the InterMED, an offshoot of the MED (Cimino 1994).

The question was posed whether Aspirin can be coated. Aspirin itself would be represented in the MED as a concept. This concept might have several descendants according to different common preparations, such as pills, drops, or capsules. Capsules consist of two parts, the active ingredient and the coating. Thus, the answer is "yes, Aspirin can be coated." In our research, we want to answer a question that a human could answer quickly in a similarly quick manner, avoiding the overhead of a general-purpose reasoner. One way to answer this question quickly within our framework would be to use *mixed transitivity reasoning* which combines different hierarchical relations into one single hierarchy. The combined hierarchy would permit a fast positive answer to the given question.

In the next section we will describe our previous work on massively parallel reasoning in detail (Lee 1993, Lee 1995, Lee 1996a, Lee 1996b). In Sections 3 and 4 we will concentrate on the definitions and algorithms necessary for intuitively correct reasoning with mixed relations and for implementing this kind of reasoning within our massively parallel paradigm. Section 5 describes experimental results of our work with the InterMED. Section 6 contains our conclusions.

2 EFFICIENT REPRESENTATIONS FOR CLASS HIERARCHIES

This research has focused on building a massively parallel transitive closure reasoner, called Hydra, that can dynamically assimilate any transitive, binary relation and efficiently answer queries using the transitive closure of all those relations (Lee 1996b). Hydra can dynamically insert new concepts or new links into a knowledge base. Hydra can respond to questions using transitive part relations (Schubert 1987), or to questions of the kind "Is an elephant bigger than a can opener"? if it knows that an elephant is bigger than a person, and a person is bigger than a can opener.

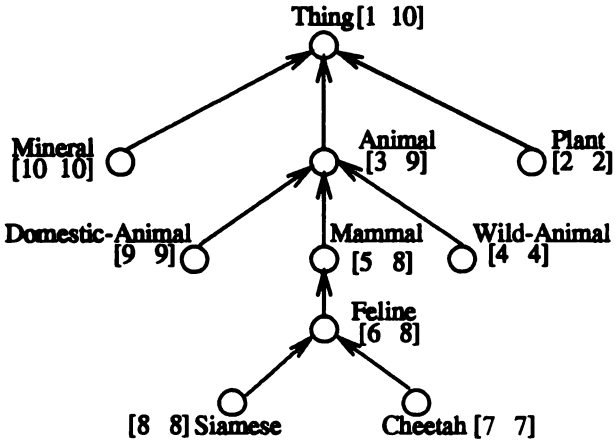


Figure 2: IS-A Hierarchy

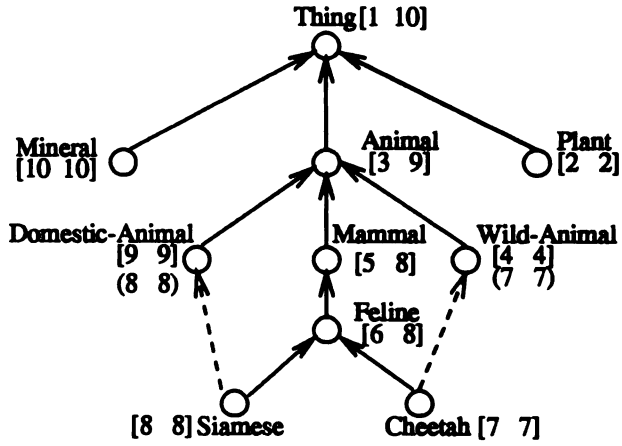


Figure 3: Directed Acyclic Graph of Class Hierarchy

Our tool of choice for achieving fast query and update operations is fine-grained parallelism. This raises the question of how to map the IS-A hierarchy onto the available space of processors. The most obvious intuitive choice is to assign every class of the hierarchy to a single processor. However, this intuitive choice does not carry over to the links between classes (Lee 1996b). We have developed a three step mapping (Figure 1) to deal with this problem.

Step 1: In the first step, an IS-A hierarchy of classes of the real world is mapped onto an isomorphic DAG of nodes, with one class per node. In the simplest possible case this hierarchy is a tree. It consists of nodes, which stand for classes, and connecting arcs, which stand for the IS-A relation. In Figure 2, an example of such an IS-A hierarchy is shown. The IS-A relation is transitive. The node Feline is connected by one IS-A arc to the node Mammal, which means that every Feline is a Mammal. Due to the transitivity of the IS-A relation, every Siamese is also a Mammal, etc.

More interesting than trees are directed acyclic graphs which open the possibility of multiple inheritance. In Figure 3, Siamese has another parent, Domestic Animal. In addition, we have also extended the representation to mixed inheritance hierarchies, i.e., hierarchies that combine relations such as IS-A, Part-of, Contained-in, Greater-than, etc., in one reasoning module.

Step 2: In the second step (Figure 1), the *hierarchy* of nodes is mapped onto a *set* of those nodes, so that every node is annotated with one or more number pairs. This is called the node set representation. The number pair assignment was developed based on Schubert *et al.*'s (Schubert 1983, Schubert 1987) special-purpose reasoner for subclass verification. Schubert *et al.*'s approach (in a tree) permits transitive closure reasoning in constant time, practically independent of the size of the knowledge base. Note that in Figure 2, we may conclude that a Cheetah is an Animal, because [7 7] is contained in [3 9]. Based on techniques that appear in (Schubert 1987, Agrawal 1988), we have extended this work to directed acyclic graphs (DAGs) (Figure 3).

In Figure 3, Siamese is a Domestic Animal because [8 8] is a subinterval (or equal to) (8 8). This is explained in more detail in the next section.

The node set representation makes it easy to represent class hierarchies on arrays of processors. The node set representation is completely order independent, i.e., node sets are used without loss of relevant hierarchy information. This simplifies the parallel update operations necessary to maintain, e.g., a class hierarchy. More details on our node set representation can be found in (Lee 1993, Lee 1996b).

Step 3: In the third step (Figure 1), this node set and the associated number pairs are mapped onto the processor space of a fine-grained parallel computer. We have developed and implemented two methods for mapping this set-based representation onto the processor space of a Connection Machine (initially CM-2, then CM-5). These two representations, the Grid Representation and the Double Strand Representation successively improve transitive closure reasoning in run time and processor space utilization.

In brief, our three step mapping (Figure 1) can be summarized as follows: *class hierarchy* → *directed acyclic*

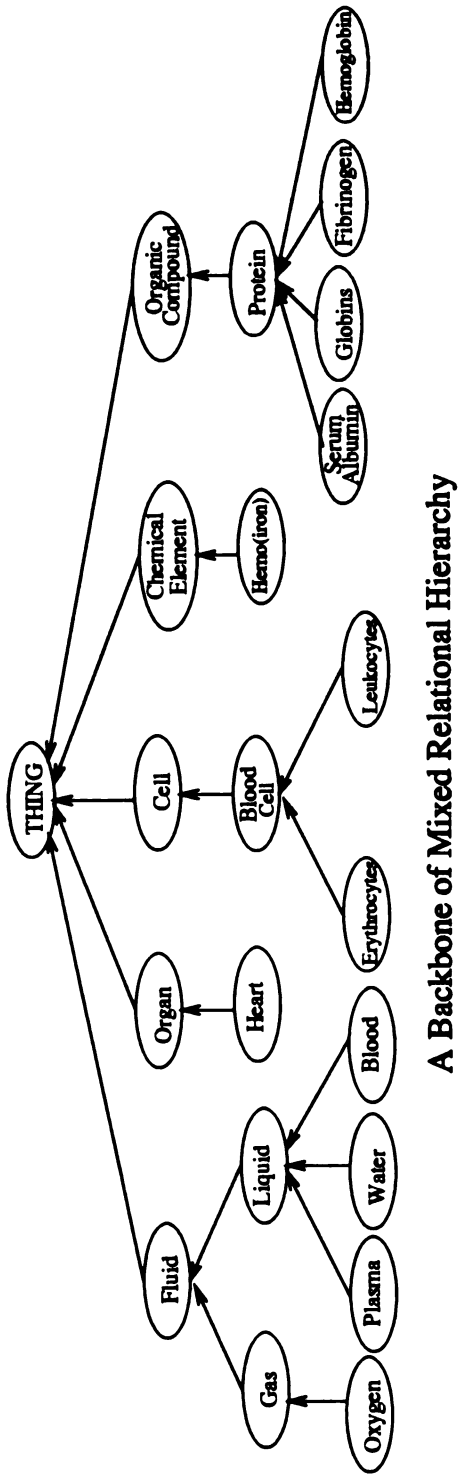


Figure 4: A Backbone of Mixed Relational Hierarchy

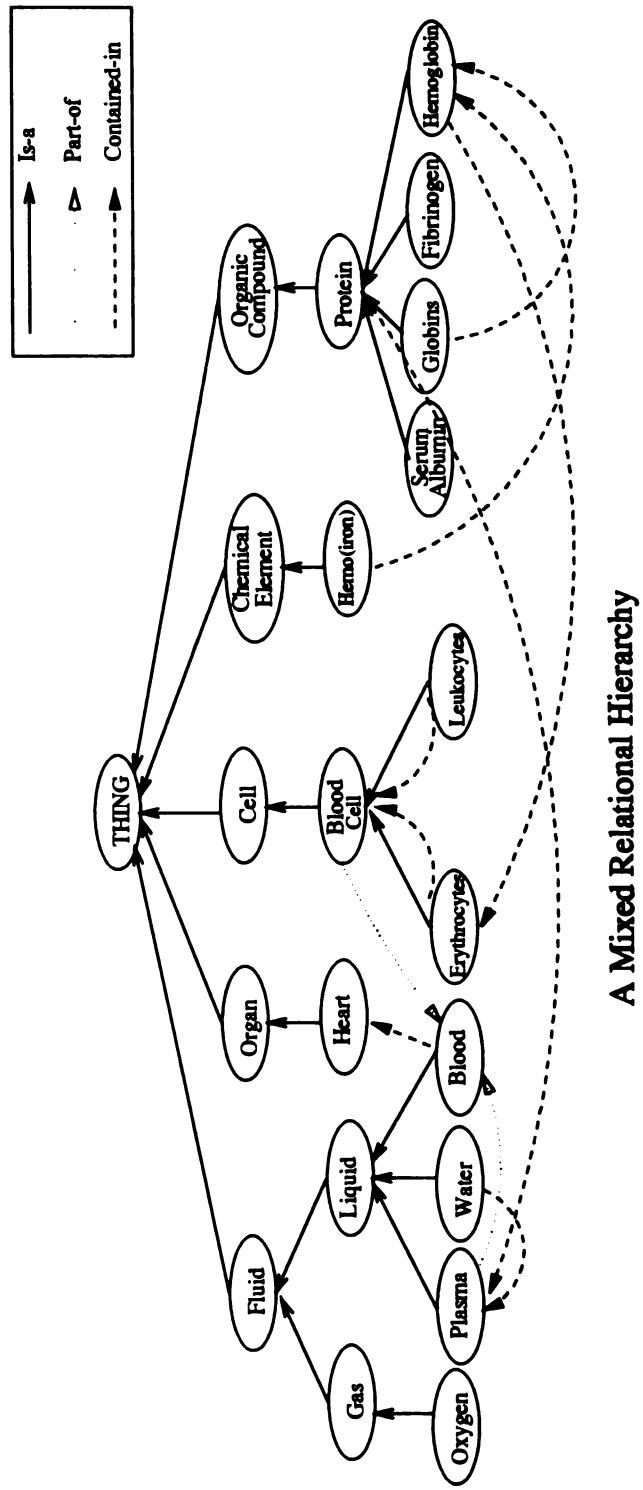


Figure 5: An Example of Mixed Relational Hierarchy

graph → node set + number pairs → processor space. We repeat that as a result of this mapping we get, within certain limitations, constant time responses for transitive closure queries (Lee 1995). In other words, it takes as much time to verify that a Cheetah is an Animal as it takes to verify that a Cheetah is a Feline (Figure 3).

3 EXTENSION TO MIXED RELATIONAL HIERARCHY

Our main idea for constructing a mixed hierarchy representation is to extend our previous approach (Lee 1995) and combine all hierarchical relations into one mixed relational hierarchy. A *mixed relational hierarchy* of the real world is mapped onto an isomorphic directed acyclic graph of nodes. Then a special spanning tree of IS-A relations becomes the backbone of the structure while other hierarchical relations form its branches. Figure 4 shows the backbone and Figure 5 shows the backbone and the branches of an example.

The following questions regarding the mixed relational hierarchy arise: First, how do we distinguish one relation from another? Second, how do we combine them when required? These questions relate not only to the construction of the hierarchy but also the transitivity reasoning. In order to avoid any possible conflict due to a combination of relations, we should design the mixed relational hierarchy to efficiently distinguish one relation from another. Let us consider the example in Figure 6. There, three relations IS-A, Part-of, and Contained-in are involved in a mixed relational hierarchy.

Our representation, extending what was described in Section 2, is generated as follows: (1) Construct an optimal spanning tree of a given DAG such that at every node with multiple parents, we select the link to the parent with the maximum number of weak predecessors. Informally, a weak predecessor Y of a node X is a predecessor which is reachable from X by an upward path containing at least one graph arc and Y is at the head of a graph arc. A graph arc is an arc that was found not to belong to the spanning tree. Assuming a top-down algorithm, all relevant graph arcs are known by the time they are needed. (2) Assign a pair of preorder and maximum number to every node. Preorder numbers are generated by a right-to-left preorder traversal of the spanning tree. The maximum number for every node is the maximum preorder number in the subtree rooted at that node. *Tree pairs* result from this step. (3) All the arcs that are not part of the optimal spanning tree are used to propagate number pairs upward. *Graph pairs* result from this step. (4) The set of all nodes together with the tree pairs and graph pairs is sufficient to represent a class hierarchy. In Figure 6 we use the notation $[\pi\ \mu]$ for *tree pairs* and the notation $(\pi\ \mu)$ for *graph pairs*. For instance, "Water is a Fluid" exactly because the number pair of Water [8 8] is contained in the pair for Fluid [5 11]. For more details, see (Lee 1993, Lee 1995, Lee 1996a). We need to integrate the different kinds of relations into our numerical representation. For this purpose, we introduce a data element, called "relation type." Each relation is associated with a relation type.

In this paper we assume that the possible relation types are s (for IS-A), p (for Part-of), and c (for Contained-in). As shown in Figure 6, we assign the lowest priority (1) to IS-A, the intermediate priority (2) to Part-of, and the highest priority (3) to Contained-in. In this assignment we follow (Winston 1987). Now we need to define rules how the relation type is assigned to a number pair.

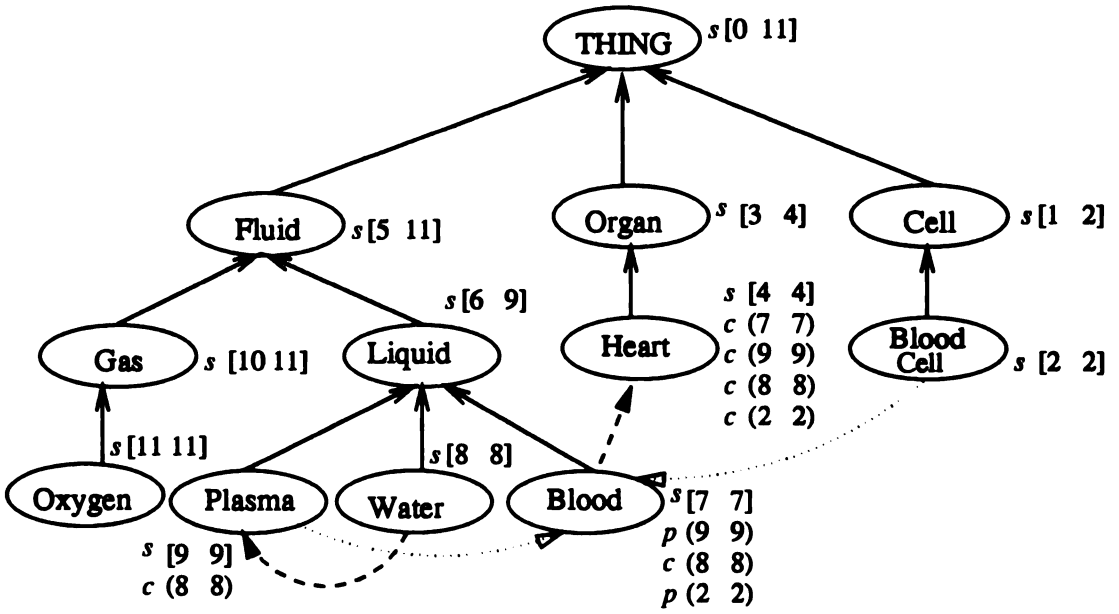
Rule 1: The relation type of a graph pair that was created by propagating a tree pair along one edge is identical to the relation type of the edge.

Rule 2: If a pair with a relation type K with relational priority X is propagated along an edge with a relation type L with relational priority Y then the result

$$R = \begin{cases} K & \text{iff } Y \leq X \\ L & \text{iff } Y > X \end{cases}$$

is the relation type of the pair at the head of the edge.

In our example (Figure 6) assume that a Part-of arc from Plasma to Blood was just inserted. Now the tree pair s[9 9] and the graph pair c(8 8) need to be propagated to the nodes (Blood, Heart) (Lee 1996a). Therefore, this pair s[9 9] is propagated through a Part-of relation from Plasma to Blood, and a Contained-in relation from Blood to Heart. The tree pair s[9 9] of Plasma is propagated through Part-of to Blood, resulting, by Rule 1, in the pair p(9 9). Continuing from Blood to Heart, the pair p(9 9) needs to be changed to c(9 9), by Rule 2. In contrast, the graph pair c(8 8) at Plasma has a Contained-in relation type and its priority is higher than the Part-of relation of the arc from Plasma to Blood and is equal to Contained-in of the arc from Blood to Heart. Therefore, the pair c(8 8) is propagated to Blood and Heart with its own relation type, by Rule 2. At this point, because Heart has a pair c(8 8) that is propagated from Water, we can conclude directly that Heart contains Water, using the



A Mixed Relational Hierarchy

Symbol	Relation	Priority	Relation Type
——→	Is-a	1 (low)	s
·····▷	Part-of	2 (mid)	p
- - - - - - - →	Contained-in	3 (high)	c

Figure 6: An Example of Constructing Mixed Relational Hierarchy

relation type *c*.

4 TRANSITIVE REASONING IN MIXED RELATIONAL HIERARCHY

In this section we show that we can achieve constant time responses (for a given machine size) for parallel transitive closure queries in a mixed hierarchy. Assume that $R_1, R_2, \ldots, R_n$ are hierarchical relations.

Definition 1: A *target of transitivity*, τ, is a node at the end (top) of a path that is used for transitive closure reasoning.

Definition 2: A *source of transitivity*, σ, is a node at the start (bottom) of the path that is used for transitive closure reasoning.

We will now define paths with two different kinds of transitivity.

Definition 3: A path P from σ to τ is *purely transitive* iff $P = \sigma\ R_1\ \sigma_1\ R_2\ \sigma_2\ \ldots\ R_n\ \tau$ and $R_1 = R_2 = \ldots = R_n$.

Definition 4: A path P from σ to τ is *mixed transitive* ($\sigma\ R^x\ \tau$) iff $P = \sigma\ R_1\ \sigma_1\ R_2\ \sigma_2\ \ldots\ R_n\ \tau$ and R^x is such that Priority(x) = Maximum(Priority(R_1), Priority(R_2) , ..., Priority(R_n)).

Both transitivities satisfy the following property: If $\sigma\ R_1\ \sigma_1\ \ldots\ \sigma_n\ R_n\ \tau$ holds, then ($\sigma\ R\ \tau$) & ($R = R_1$ *or* $\ldots\ R = R_n$). Importantly, pure transitivity reasoning and mixed transitivity reasoning can be done in one step. We will present how these mechanisms can be integrated during reasoning.

We now query which relation holds form σ to τ. Our

transitive reasoning mechanism works based on an extension of a number pair propagation algorithm introduced in (Lee 1996a). There we proved that if the relation type of the path is IS-A, i.e., there is a tree path from σ to τ, we can achieve the effect of having all graph pairs of σ at τ, without actually propagating these pairs to τ, resulting in an additional saving of space. Above "achieve the effect of having all graph pairs" means that we can perform constant time subclass verification and all operations that rely on subclass verification, including propagation itself. We now have to prove that the algorithm still works after including the concept of relation type.

In Figure 7, the left part shows propagation based on Agrawal's approach (Agrawal 1989) and the right part shows propagation based on our approach. When a graph arc is inserted from Ω to σ with the relation type x, we propagate the pair $x(\pi\ \mu)$ only to σ instead of propagating $x(\pi\ \mu)$ up to all tree predecessors of σ (including τ). Our approach is called Maximally Reduced Propagation and its advantages are explained in (Lee 1996a). This example shows how we reduce the number of graph pairs for all tree predecessors of σ. That the relevant algorithms work was proved in (Lee 1996a). We now show how we can achieve constant time mixed transitivity reasoning for the x relation type from Ω to all tree predecessors of σ (including σ and τ) with the maximally reduced pair propagation.

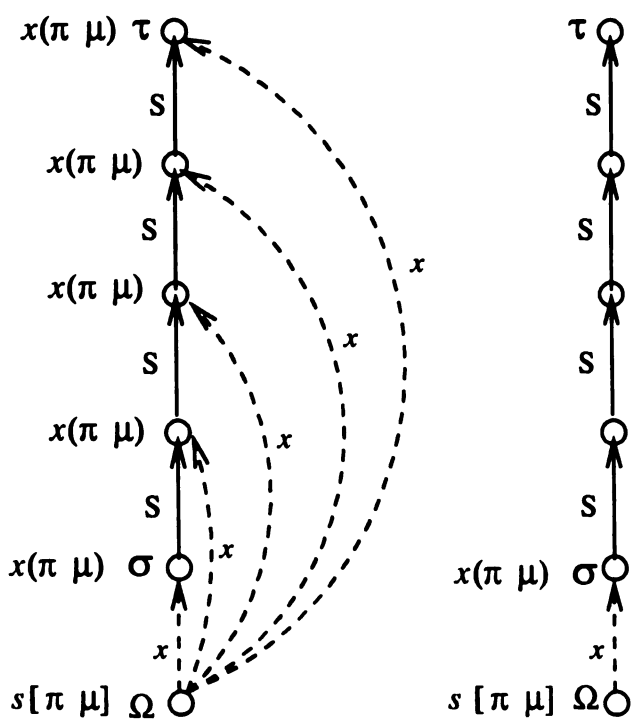


Figure 7: Maximally Reduced Propagation

Lemma 1: Pure Transitivity in IS-A Path
Let R^x be an IS-A relation ($R^x = R^s$). If $\sigma\ R^x\ \tau$ holds through a pure inference path, then after applying our propagation algorithm from (Lee 1996a), the target τ or one of its tree successors will have a number pair with the relation type $x = s$ propagated from σ or from one of its tree predecessors.

Proof: By contradiction, assume that the target τ has a graph pair $s(\pi_s\ \mu_s)$ from the source σ, although an arc A in the inference path from σ to τ is not an IS-A relation. Since the IS-A relation has the lowest relational priority among all relations, the pair $s(\pi_s\ \mu_s)$ can not be propagated through A unless the relation type of $s(\pi_s\ \mu_s)$ changes to the relation type of the arc A (Rule 2). Therefore, the pair $(\pi_s\ \mu_s)$ cannot be associated with the relation type $x = s$. This results in a contradiction. ■

Lemma 2: Mixed Transitivity
Let R^x be a relation. If a source σ relates by R^x to a target τ because of a mixed inference path from σ to τ, the target τ or its tree successor must have a number pair with relation type x propagated from the source σ or its tree predecessor.

Proof: By using Rule 2, this is trivial. ■

Theorem 1: If τ (or a tree successor of τ) has a pair that contains (or is equal to) a pair from σ (or a tree predecessor of σ) then the relation type of the pair of τ (or a tree successor of τ) tells the relation which actually holds between σ and τ.

Proof: We prove it by using the above two lemmas. If the relation is an IS-A relation, the query is an instance of IS-A pure transitivity reasoning (Lemma 1). Otherwise, the query is an instance of mixed transitivity reasoning or pure transitivity reasoning with other relations (Lemma 2). ■

Now we will show how to answer mixed and pure transitivity queries within our paradigm in parallel. We divide pure transitivity into two subcases: one for an IS-A relation and another for other hierarchical relations. A formal description of pure transitivity with other relations will be omitted for space reasons. For more details, see (Lee 1996b). Our massively parallel Double Strand Representation (Section 2) uses pairs of adjacent processors to represent a sequence of graph pairs. In each pair one processor has an odd processor ID and is used to represent a node which propagates its tree pair and its right adjacent processor has an even processor ID and is used to represent a node from which a number pair is propagated (Figure 8).

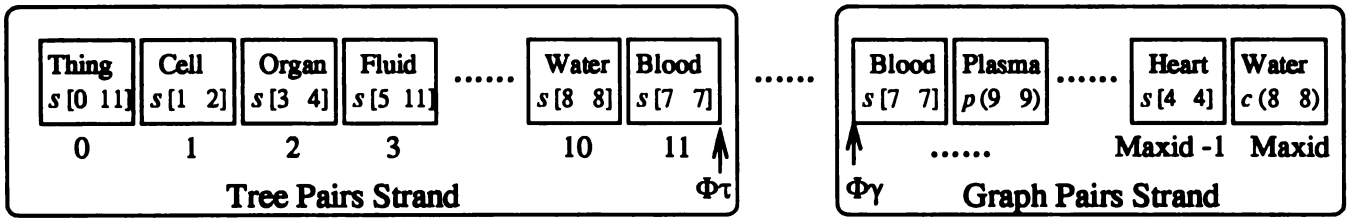


Figure 8: Double Strand Representation

We now introduce some CM-5 terminology. A parallel variable (*pvar*) is an array (possibly multidimensional) where every component is maintained by its own processor and all values are usually changed in the same way and in parallel (TMC 1988).

In the algorithm, variables marked with !! are parallel variables, and operations marked with !! or involving parallel variables are parallel operations. The parallel variable pre!! contains for every node its preorder number, and the expression max!! stands for a parallel variable that contains for every node its maximum number, and the expression reltype!! stands for a parallel variable that contains for every number pair its relation type. The functions prenum(), maxnum(), and tree-pair() retrieve the preorder number, maximum number, and the tree pair, respectively, for the given argument.

Additionally, the variable Φ_γ represents the lower bound of the graph pairs strand and Φ_τ represents the upper bound of the tree pairs strand. The parallel function *self-address!!* returns IDs of all active processors and *oddp!!* contains TRUE on a processor if the processor's ID is an odd number. The parallel control structure ACTIVATE-PROCESSORS-WITH consists of two parts. The first part describes a set of processors to be activated. The second part, starting with DO, describes what operations should be performed on all active processors.

Algorithm: Pure Transitivity with IS-A (σ, τ)

- Activate every processor that contains a pair with the IS-A relation type (s).
- (Case 1: a tree path from σ to τ)
  Among active processors, check whether the tree pair of σ is contained in or equal to the tree pair of τ.
- (Case 2: a graph path from σ to τ)
  Among active processors, check whether any processor has a tree pair of τ or a pair of a tree successor of τ at an odd processor ID = x, and the processor with ID = $x + 1$ contains a pair [propagated] from the tree predecessor of σ.
- Iff Case 1 or Case 2 is the case, return "yes."

We now show a function IS-A-VERIFY that performs pure subclass verification. As we mentioned above, if τ is a tree predecessor of σ (by IS-A-VERIFY-1) or τ is a graph predecessor of σ (by IS-A-VERIFY-2), then σ IS-A τ. Note that as every tree pair has associated with it a single relation type s, it is not necessary to check the relation type for IS-A-VERIFY-1.

```
; B is-a A iff IS-A-VERIFY returns TRUE.
IS-A-VERIFY (σ, τ: Node): BOOLEAN
  return(IS-A-VERIFY-1(σ, τ) OR
         IS-A-VERIFY-2(σ, τ))


IS-A-VERIFY-1 (σ, τ: Node): BOOLEAN
  ; If τ is a tree predecessor of σ, then
  ; the tree pair of τ subsumes the tree
  ; pair of σ.
  ACTIVATE-PROCESSORS-WITH
      prenum(tree-pair(σ)) ≥!!
          prenum(tree-pair(τ)) AND!!
      maxnum(tree-pair(σ))≤!!
          maxnum(tree-pair(τ)) AND!!
      self-address!!() ≤!! Φτ
  DO BEGIN
    IF any processor is still active THEN
      return TRUE
  END
```

Now we will show how to verify that σ IS-A τ when τ is a graph predecessor of σ. Remember that a pair of processors (U, V) in the graph pairs strand is used to represent a graph pair propagation. The tree pair in the odd processor (U) is used to represent a node τ and the graph pair in the even processor (V) is used to represent a node which propagates its tree pair to τ. Therefore, we are looking for a pair of processors (U, V) such that the tree pair of τ or of one of its

tree successors is contained in processor U and the graph pair of σ or its tree predecessor is contained in processor V. In the following functions the expression $mark!![x] := y$ means that the pvar *mark!!* on the processor with the ID x is assigned the value y. We omit the initialization of *mark!!*.

```
IS-A-VERIFY-2 (σ, τ: Node): BOOLEAN
  ; Activate every occurrence of the tree pair of
  ; τ or its tree successors associated with the
  ; s relation type in the graph pairs strand. Set
  ; the parallel flag mark!! on the right neighbor
  ; processors of the active processors.
  ACTIVATE-PROCESSORS-WITH
      reltype!! =!! s AND!!
      pre!! ≥!! prenum(tree-pair(τ)) AND!!
      max!! ≤!! maxnum(tree-pair(τ)) AND!!
      self-address!!() ≥!! Φγ AND!!
      oddp!! (self-address!!())
  DO BEGIN
      mark!![self-address!!() +!! 1]:= 1
  END

  ; Test whether any marked processor has the tree
  ; pair with the relation type s from σ or from
  ; a tree predecessor of σ, as a graph pair. If this
  ; is the case, return TRUE.
  ACTIVATE-PROCESSORS-WITH
      reltype!! =!! s AND!!
      pre!! ≤!! prenum(tree-pair(σ)) AND!!
      max!! ≥!! maxnum(tree-pair(σ)) AND!!
      mark!![self-address!!()] =!! 1
  DO BEGIN
      IF any processor is still active THEN
         return TRUE
  END


MIXED-RELATION-VERIFY
  (ξ: Relation Type; σ, τ: Node): BOOLEAN
  ; Activate every occurrence of the tree pair of τ
  ; and of its tree successors associated with the
  ; relation type s. Set the parallel flag mark!!
  ; on the right neighbor processors of the active
  ; processors.
  ACTIVATE-PROCESSORS-WITH
      reltype!! =!! s AND!!
      pre!! ≥!! prenum(tree-pair(τ)) AND!!
      max!! ≤!! maxnum(tree-pair(τ)) AND!!
      self-address!!() ≥!! Φγ AND!!
      oddp!! (self-address!!())
  DO BEGIN
      mark!![self-address!!() +!! 1]:= 1
  END

  ; Test whether any marked processor has the tree
  ; pair from σ, or from a tree predecessor of σ,
  ; as a graph pair with the relation type ξ. If this
  ; is the case, return TRUE.
  ACTIVATE-PROCESSORS-WITH
      reltype!! =!! ξ AND!!
      pre!! ≤!! prenum(tree-pair(σ)) AND!!
      max!! ≥!! maxnum(tree-pair(σ)) AND!!
      mark!![self-address!!()] =!! 1
  DO BEGIN
      IF any processor is still active THEN
         return TRUE
  END
```

Consider again our example of pure transitivity in Figure 6: Is Water a Fluid? The tree pair of Fluid s[5 11] contains the tree pair of Water s[8 11]. The answer "yes" can be given by comparing these two tree pairs (by IS-A-VERIFY). What about the mixed transitivity example? Is Water contained in Organ? As the query is about Contained-in (c) and σ is Water and τ is Organ, the procedure MIXED-RELATION-VERIFY will be invoked with a list of arguments (c, Water, Organ). We are first looking for Organ and its tree successors. These are nodes with tree pairs contained in s[3 4]. However, as we are interested in propagations, we are looking for these tree successors (or Organ itself) in the graph pairs strand. There we find Heart s[4 4] with a right neighbor Water c(8 8) (Figure 8). In the second stage we are looking for a tree predecessors of Water s[8 8] (or Water itself), but with s replaced by the value of ξ, which is c. This perfectly matches the pair c(8 8) identified in the first step, and we can conclude that the answer is "yes, Water is contained in Organ." Due to parallel processing, the mixed transitive closure query can be answered in constant time.

An analysis of the parallel operations involved shows that both kinds of queries can be answered with our parallel representation in constant time, independent of the size of the knowledge base (assuming constant machine size) (Wang 1989).

5 EXPERIMENTS WITH InterMED

We have performed a set of experiments that analyze the run-times for purely transitive IS-A queries and mixed relational queries using data from the InterMED. For this experiment, we have used 2495 nodes, 3372 IS-A links, and 682 Pharmaceutic-component-of links from the InterMED. Note that the

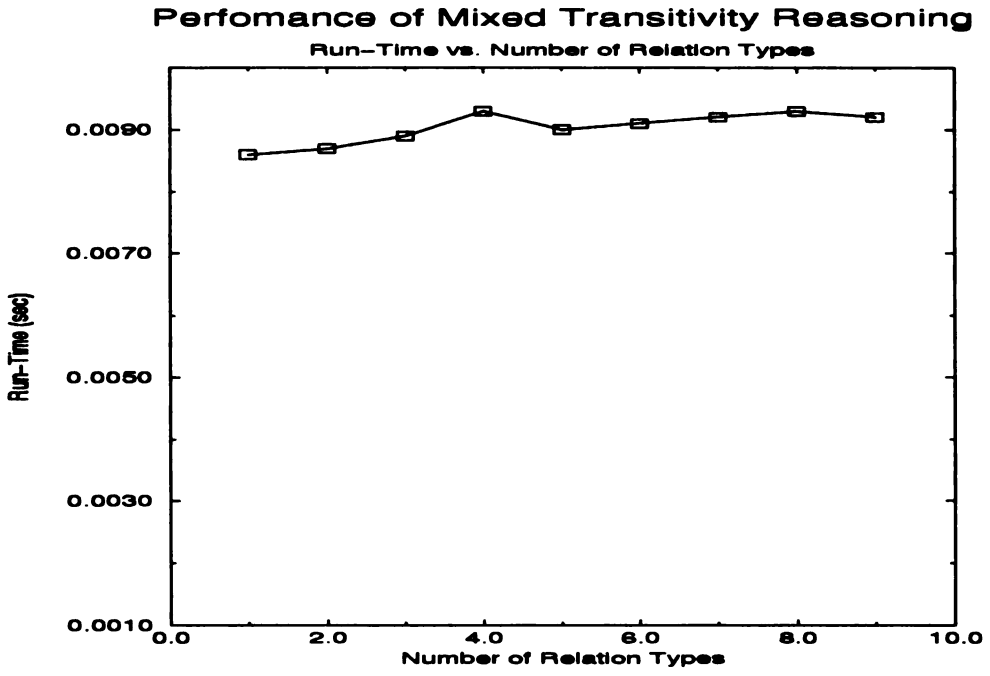


Figure 9: An Experiment of Mixed Transitivity Reasoning

fan-in and fan-out of the hierarchy are not of immediate experimental importance because the node set representation eliminates the explicit IS-A links. The only relevant factor is the number of graph pairs generated. The experiments were performed on a Connection Machine CM-5 (TMC 1988) programmed in *LISP. For this purpose, we constructed a hierarchy for each experiment as follows: (1) We created an IS-A hierarchy (SubClass Hierarchy, SCH); (2) We created a mixed relational hierarchy (Mixed Relational Hierarchy, MRH).

The results in Table 1 show that the run-times for transitive reasoning in both hierarchies are the same within unit processor space (8K virtual processors). We show run-times for "normal" transitive reasoning in the SCH hierarchy, for pure transitivity reasoning in the MRH hierarchy, and for mixed transitivity reasoning. Specifically, IS-A-Verify-1 and IS-A-Verify-2 represent the first and the second cases of the parallel pure transitivity reasoning algorithm in Section 4.

As another experiment, we measured how the number of relations is related to the run-time of mixed relational queries. For this experiment, we tested the run-times by increasing the number of relations in transitive closure queries within constant processor space (8K). These additional relation types were created by a random generator. Figure 9 shows that the run-time for mixed relational transitivity reasoning is independent of the number of relations in a hierarchy. In summary, we can conclude that our mixed transitive queries can be executed in constant time, as in a pure IS-A hierarchy.

6 CONCLUSIONS

In this paper, we have discussed techniques for fast evaluation of transitive queries in mixed relational hierarchies. We have introduced a paradigm based on number pair propagation with a relation type. This paradigm avoids any invalid conclusions of mixed relational transitivity and integrates several relations when needed. Due to our new mixed relational representation and parallel processing, it is possible to perform fast mixed transitivity reasoning. Experimental results using an existing medical vocabulary show that mixed transitivity reasoning can be executed in constant time assuming constant processor space.

Acknowledgements

This work was conducted using the computational resources of the Northeast Parallel Architectures Center (NPAC) at Syracuse University, which is funded

Table 1: Experimental Results for Three Approaches

Hierarchy Type	*Reasoning Type*		*Run-Time*
SCH	*Single Transitivity Reasoning*	*IS-A-Verify-1*	0.00042 (sec)
		IS-A-Verify-2	0.0082 (sec)
MRH	*Pure Transitivity Reasoning*	*IS-A-Verify-1*	0.00042 (sec)
		IS-A-Verify-2	0.0084 (sec)
	Mixed Transitivity Reasoning		0.0086 (sec)

by and operates under contract to DARPA and the Air Force Systems Command, Rome Air Development Center (RADC), Griffiss Air Force Base, NY, under contract# F306002-88-C-0031.

This research was (partially) done under a cooperative agreement between the National Institute of Standards and Technology Advanced Technology Program (under the HIIT contract, number 70NANB5H1011) and the Healthcare Open Systems and Trials, Inc consortium.

References

R. Agrawal, A. Borgida, and H. V. Jagadish. Efficient management of transitive relationships in large data and knowledge bases. In *Proceedings of the 1989 ACM SIGMOD International Conference on the Management of Data*, pages 253–262, Portland, OR, 1989.

R. Agrawal, S. Dar, and H. V. Jagadish. Direct transitive closure algorithms: Design and performance evaluation. *ACM Transactions on Database Systems*, 15(3):428–458, 1990.

J. J. Cimino, A. A. Aguirre, S. B. Johnson, and P. Peng. Generic queries for meeting clinical information needs. *Bulletin of the Medical Library Association*, 81(2):195–206, 1993.

J. J. Cimino, P. D. Clayton, G. Hripcsak, and S. B. Johnson. Knowledge-based approaches to the maintenance of a large controlled medical terminology. *Journal of the American Medical Informatics Association*, 1(1):35–50, 1994.

J. J. Cimino, P. L. Elkin, and G. O. Barnett. As we may think: The concept space and medical hypertext. *Computers and Biomedical Research*, 25:238–263, 1992.

J. J. Cimino, G. Hripcsak, S. B. Johnson, and P. D. Clayton. Designing an introspective, multipurpose, controlled medical vocabulary. In *Proceedings of the Thirteenth Annual Symposium on Computer Applications in Medical Care*, pages 513–518. IEEE Computer Society Press, Los Alamitos, CA, 1989.

Paul R. Cohen and Cynthia L. Loiselle. Beyond isa: Structures for plausible inference in semantic networks. In *Proceedings of AAAI-88*, pages 415–420, Saint Paul, Minnesota, 1988.

M. P. Evett, W. A. Andersen, and J. A. Hendler. Massively parallel support for efficient knowledge representation. In *Proc. of the 13th Int. Joint Conference on Artificial Intelligence*, pages 1325–1330. Morgan Kaufmann, San Mateo, CA, 1993.

M. P. Evett, J. A. Hendler, and W. A. Andersen. Massively parallel support for computationally effective recognition queries. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 297–302. MIT Press, Cambridge, MA, 1993.

M. Evett, L. Spector, and J. Hendler. Knowledge representation on the connection machine. In *Supercomputing '89*, pages 283–293, Reno, Nevada, 1989.

J. Geller and C. Y. Du. Parallel implementation of a class reasoner. *Journal of Experimental and Theoretical Artificial Intelligence*, 3:109–127, 1991.

J. Geller. Upward-inductive inheritance and constant time downward inheritance in massively parallel knowledge representation. In *Proceedings of the Workshop on Parallel Processing for AI at IJCAI 1991*, pages 63–68, Sydney, Australia, 1991.

J. Geller. Innovative applications of massive parallelism. *AAAI 1993 Spring Symposium Series Reports, AI Magazine*, 14(3):36, 1993.

J. Geller. Massively parallel knowledge representation. In *AAAI Spring Symposium Series Working Notes: Innovative Applications of Massive Parallelism*, pages 90–97, 1993.

J. Geller. Advanced update operations in massively parallel knowledge representation. In H. Kitano and J. Hendler, editors, *Massively Parallel Artificial Intelligence*. AAAI/MIT Press, 1994, forthcoming.

J. Geller. Inheritance operations in massively parallel knowledge representation. In L. Kanal, V. Kumar, H. Kitano, and C. Suttner, editors, *Parallel Processing for Artificial Intelligence*. Elsevier Science Publishers,

Amsterdam, 1994, forthcoming.

M. Halper, J. Geller, and Y. Perl. An OODB "part" relationship model. In Yelena Yesha, editor, *Proceedings of the First International Conference on Information and Knowledge Management*, pages 602–611, Baltimore, MD, nov 1992.

M. Halper. *A Comprehensive Part Model and Graphical Schema Representation for Object-Oriented Databases.* PhD dissertation, CIS Department, New Jersey Institute of Technology, 1993.

M. Halper, J. Geller, and Y. Perl. Value propagation in object-oriented database part hierarchies. In *Proceedings of the 2nd Int'l Conference on Information and Knowledge Management*, pages 606–614. Washington, DC, 1993.

M. Halper, J. Geller, and W. Klas. Integrating a part relationship into an open oodb system using metaclasses. In *Proceedings of the 3rd Int'l Conference on Information and Knowledge Management*, pages 10–17, Gaithersburg, MD, 1994.

James Hendler, Jaime Carbonell, Douglas Lenat, Richiro Mizoguchi, and Paul Rosenbloom. Very large knowledge bases–architecture vs engineering. In *Proceedings of IJCAI 1995*, pages 2033–2037, Montreal, Quebec, 1995.

H. Kitano and T. Higuchi. High performance memory-based translation on IXM2 massively parallel associative memory processor. In *Proceedings of IJCAI 1991*, pages 149–154, Sydney, Australia, 1991.

H. Kitano and T. Higuchi. Massively parallel memory-based parsing. In *Proceedings of IJCAI 1991*, pages 918–924, Sydney, Australia, 1991.

H. Kitano. Massively parallel AI and its application to natural language processing. In *Proceedings of the Workshop on Parallel Processing for AI at IJCAI 1991*, pages 99–105, Sydney, Australia, 1991.

H. Kitano, D. Moldovan, and S. Cha. High performance natural language processing on semantic network array processor. In *Proceedings of IJCAI 1991*, pages 911–917, Sydney, Australia, 1991.

E. Y. Lee and J. Geller. Representing transitive relationships with parallel node sets. In Bharat Bhargava, editor, In *Proceedings of the IEEE Workshop on Advances in Parallel and Distributed Systems*, pages 140–145. IEEE Computer Society Press, Los Alamitos, CA, 1993.

E. Y. Lee and J. Geller. Parallel operations on class hierarchies with double strand representations. In *IJCAI-95 Workshop Program Working Notes*, pages 132–142, Montreal, Quebec, 1995.

E. Y. Lee and J. Geller. Constant time inheritance with parallel tree covers. In *Proceedings of the Ninth Florida AI Research Symposium*, pages 243–250, Key West, FL, 1996.

E. Y. Lee. Massively parallel reasoning in transitive relationship hierarchies *PhD dissertation*, CIS Department, New Jersey Institute of Technology, 1996.

M. R. Quillian. Semantic memory. In M. L. Minsky, editor, *Semantic Information Processing*, pages 227–270. The MIT Press, Cambridge, MA, 1968.

L. Shastri and V. Ajjanagadde. An optimally efficient limited inference system. In *Proceedings of IJCAI-90*, pages 563–570, Boston, MA, 1990.

L. K. Schubert. Problems with parts. In *Proc. of the 6th International Joint Conference on Artificial Intelligence*, pages 778–784. Morgan Kaufmann Publishers, Los Altos, CA, 1979.

L. Shastri. *Semantic Networks: an Evidential Formalization and its Connectionist Realization.* Morgan Kaufmann Publishers, San Mateo, CA, 1988.

L. Shastri. Default reasoning in semantic networks: a formalization of recognition and inheritance. *Artificial Intelligence*, 39(3):283–356, 1989.

L. K. Schubert, M. A. Papalaskaris, and J. Taugher. Determining type part, color, and time relationships. *Computer*, 16(10):53–60, 1983.

L. K. Schubert, M. A. Papalaskaris, and J. Taugher. Accelerating deductive inference: special methods for taxonomies colors and times. In N. Cercone and G. McCalla, editors, *The Knowledge Frontier*, pages 187–220. Springer Verlag, New York, NY, 1987.

Thinking Machines Corporation. **LISP Reference Manual Version 5.0 edition.* Thinking Machines Corporation, Cambridge, MA, 1988.

D. L. Waltz. Massively parallel AI. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 1117–1122. Morgan Kaufmann, San Mateo, CA, 1990.

W. Wang, S. Iyengar, and L. M. Patnaik. Memory-based reasoning approach for pattern recognition of binary images. *Pattern Recognition*, vol. 22, no 5, pp. 505–518, 1989.

M. E. Winston, R. Chaffin, and D. Herrmann. A taxonomy of part-whole relations. *Cognitive Science*, 11(4):417–444, 1987.

DLMS: An Evaluation of KL-ONE in the Automobile Industry

Nestor Rychtyckyj
Ford Motor Company
Manufacturing Quality Business Systems
P.O. Box 1586, Room B154
Dearborn MI, 48121
usfmc53B@ibmmail.com

Abstract

Ford Motor Company's Direct Labor Management System (DLMS) utilizes a knowledge representation scheme based on the KL-ONE family of languages to represent the world of automobile vehicle assembly knowledge. DLMS is an implemented system that has been utilized by Ford personnel since 1991 and it remains an integral part of Ford's worldwide manufacturing strategy. This paper will describe the rationale for the use of KL-ONE, provide an overview of how KL-ONE was implemented into DLMS, discuss the advantages and disadvantages of using a knowledge representation scheme such as KL-ONE in a production environment and present an evaluation of how knowledge representation and reasoning systems can be successfully utilized in industry.

INTRODUCTION

Ford Motor Company's Direct Labor Management System (DLMS) is the knowledge-based subsystem of a complex multiphase manufacturing process planning system. Since its original deployment in 1991, DLMS has been utilized by hundreds of users throughout Ford's automobile and truck assembly plants in North America. Currently DLMS is being expanded to Ford's assembly plants around the world. The knowledge that is used to drive the manufacturing assembly process in DLMS is stored in a KL-ONE knowledge representation scheme.

This paper will discuss the long-term implications of utilizing KL-ONE in a dynamic environment such as automobile assembly planning. Issues such as knowledge base validation and verification, maintenance and adaptability to changing market conditions will also be discussed.

DLMS OVERVIEW

The Direct Labor Management System (DLMS) is an implemented system utilized by Ford Motor Company's Vehicle Operations division to manage the use of labor on the assembly lines throughout Ford's vehicle assembly plants. DLMS was designed to improve the assembly process planning activity at Ford by achieving standardization within the vehicle process build description and to provide a tool for accurately estimating the labor time required to perform the actual vehicle assembly. In addition, DLMS provides the framework for allocating the required work among various operators at the plant and builds a foundation for automated machine translation of the process descriptions into foreign languages.

The standard process planning document known as a process sheet is the primary vehicle for conveying the assembly information from the initial process planning activity to the assembly plant. A process sheet contains the detailed instructions needed to build a portion of a vehicle. A single vehicle may require thousands of process sheets to describe its assembly. The process sheet is written by an engineer utilizing a restricted

subset of English known as SLANG (Standard LANGuage). Standard Language allows an engineer to write clear and concise assembly instructions that are machine readable.

Figure 1 shows a portion of a process sheet written in Standard Language. This process sheet is written by an engineer at the Vehicle Operations General Office; it is then sent to the DLMS system to be "validated" before it can be released to the assembly plants. Validation includes the following: checking the process sheet for errors, generating the sequence of steps that a worker at the assembly plant must perform in order to accomplish this task and calculating the length of time that this task will require. The DLMS system interprets these instructions and generates a list of detailed actions, known as allocatable elements, that are required to implement these instructions at the assembly plant level. These allocatable elements are associated with MODAPTS (MODular Arrangement of Predetermined Time Standards) codes that are used to calculate the time required to perform these actions.

MODAPTS codes are utilized as a means of measuring the body movements that are required to perform a physical action and have been accepted as a valid work measurement system around the world. [IES 88]. For example the MODAPTS code for moving a small object with only a hand is M2; utilizing the arm gives a code of M3. The MODAPTS codes are then combined to describe a entire sequence of actions. MODAPTS codes are then converted into an equivalent time, which is needed to perform that action. Figure 2 shows the output generated by the DLMS system including a description of each action with its associated MODAPTS code.

The work instructions generated by DLMS are known as allocatable elements. These allocatable elements are then used by the engineering personnel at the assembly plant to allocate the required work among the available personnel. DLMS is a powerful tool because it provides timely information about the amount of direct labor that is required to assemble each vehicle, as well as pointing out inefficiencies in the assembly process. A more complete description of the DLMS system can be found in [O'Brien et al. 89].

Process Sheet Written in Standard Language

```
TITLE: ASSEMBLE IMMERSION HEATER TO ENGINE

10  OBTAIN ENGINE BLOCK HEATER ASSEMBLY
FROM STOCK
20  LOOSEN HEATER ASSEMBLY TURNSCREW USING
POWER TOOL
30  APPLY GREASE TO RUBBER O-RING AND CORE
OPENING
40  INSERT HEATER ASSEMBLY INTO RIGHT REAR
CORE PLUG HOSE
50  ALIGN SCREW HEAD TO TOP OF HEATER
60  SEAT ASSEMBLY UNTIL ENGINE BLOCK AND
ASSEMBLY ARE FLUSH
70  SECURE SCREW USING POWER TOOL TO TIGHTEN
HEATER ASSEMBLY
TOOL 20 1 P AAPTCA TSEQ RT ANGLE NUTRUNNER
TOOL 30 1 C COMM TSEQ GREASE BRUSH
```

Figure 1.

Resulting Work Instructions Generated by DLMS

```
LOOSEN HEATER ASSEMBLY TURNSCREW USING
POWER TOOL
GRASP POWER TOOL (RT ANGLE NUTRUNNER)
<01M4G1>
POSITION POWER TOOL (RT ANGLE NUTRUNNER)
<01M4P2>
ACTIVATE POWER TOOL (RT ANGLE NUTRUNNER)
<01M1P0>
REMOVE POWER TOOL (RT ANGLE NUTRUNNER)
<01M4P0>
RELEASE POWER TOOL (RT ANGLE NUTRUNNER)
<01M4P0>
```

Figure 2.

The DLMS system consists of five main subsystems: parser, analyzer, simulator, knowledge base manager, and the error checker. The input into DLMS is a process sheet; it is initially parsed to break down the sentence into its lexical components which includes the verb, subject, modifiers, prepositional phrases and other parts of speech. Since Standard Language is a restricted subset of English, the parser has a very high rate of success in properly parsing the input from the process sheets. The parser utilizes the Augmented Transition Network (ATN) method of parsing [Charniak, et. al 87]. Any process element that is not parsed successfully will then be flagged by one of the error rules that will (hopefully) suggest to the user how to correct this element. The analyzer will then use the components of the parsed element to search the knowledge base (or taxonomy) for relevant information describing that item.

For example, the input element contained the term "HAMMER". This term will then be searched for in the taxonomy; when it is found the system will then learn all of the attributes that "HAMMER" has: (it is a Tool, its size is medium, it can be used with one hand, etc.) The system will perform this analysis on all of the components of the input element in order to select what work instructions are required.

The work instructions are then found in the taxonomy based on all of the available input and are passed on to the simulator. The simulator will use the taxonomy to generate the sequence of instructions and MODAPTS codes that will describe the input element. These work instructions will then be sent back to the user. The knowledge base manager is used to maintain the knowledge base; this maintenance may be performed by the user community or by the system developers.

All of the associated knowledge about Standard Language, tools, parts and everything else associated with the automobile assembly process is contained in the DLMS knowledge base or taxonomy. This knowledge base structure is derived from the KL-ONE family of semantic network structures and is the integral component in the success of DLMS. DLMS also contains a rulebase of over 350 rules that are used to drive the validation process and perform error-checking on the Standard Language input. DLMS was implemented in Common LISP and ART (Automating Reasoning Tool from Brightware Corporation) on the Texas Instrument Explorer platform. It is currently being ported to the Hewlett Packard UNIX platform. Figure 3 describes the current DLMS architecture.

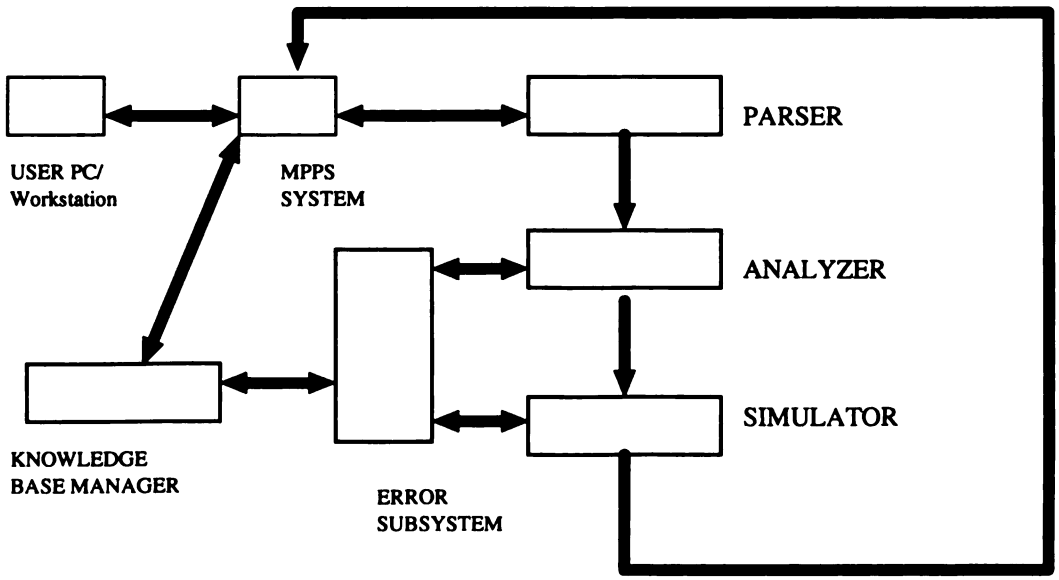


Figure 3: Current DLMS System Architecture

KL-ONE OVERVIEW

The KL-ONE knowledge representation system [Brachman, Schmolze 85] was first developed at Bolt, Baranek and Newman in the late 1970's as an outgrowth of semantic net formalisms. KL-ONE was selected for use on the DLMS project because of its adaptability for many diverse applications as well as the power of the KL-ONE classification algorithm. KL-ONE is derived from research done on semantic networks.

The principal unit of information is the "concept". Each concept has a set of components or attributes that is true for each member of the set denoted by that concept. The main form of relation between concepts is called "subsumption". Subsumption is the property by which concept A subsumes concept B if, and only if, the set denoted by concept A includes the set denoted by concept B. The KL-ONE knowledge base as used in DLMS can be described as a network of concepts with the general concepts being closer to the root of the tree and the more specific concepts being the leaves of the tree. A concept in a KL-ONE knowledge base inherits attributes from the concepts that subsume it. The power of the KL-ONE system lies in the classification scheme. The system will place a new concept into its appropriate place in the taxonomy by utilizing the subsumption relation on the concept's attributes. A much more detailed description of the KL-ONE classification scheme can be found in [Shmolze, Lipkis 81].

KL-ONE USAGE BACKGROUND

The requirement for a system to automate process planning in automobile assembly at Ford Motor Company was very evident since the early 1980's. Previously, process sheets were written in free-form English and then sent to the assembly plants for implementation. The quality and correctness of process sheets differed greatly based upon which engineer had written a particular sheet. There was no standardization with similar process sheets and industrial engineers at the assembly plants would be forced to implement work instructions based on differing process sheets. The process sheets could not describe the amount of labor required and the assembly plants were not able to accurately plan for labor requirements. Work usage instructions were written manually and the time required to accomplish a particular job would have to be measured manually.

These manual "stopwatch" time studies suffered from several major disadvantages. A time study consisted of an industrial engineer watching an assembly line worker doing their job and measuring how long each job would take. These measurements would vary from worker to worker, so that multiple time studies were required for each particular job. Since there may be hundreds or even thousands of jobs in an assembly plant, the time studies were very expensive and time-consuming. Time studies also have a very adverse effect on worker morale and are a source of resentment among the assembly personnel.

Since labor is a very significant portion of the cost of producing an automobile, there was a very strong incentive to develop a system that could both standardize the process sheet and create a tool for automatically generating work instructions and times from these process sheets. The first attempts to create DLMS were done utilizing standard third generation programming languages (COBOL) and existing IBM mainframe databases (IMS). The sheer complexity of the knowledge required to accurately generate reliable work instructions could not be represented in either a database or in a program. A database could easily store the amount of data required, but the relationships between the various components in the database could not be adequately represented. A program could be written that could explicitly list all of the inputs and desired outputs, but this program would quickly become obsolete and be impossible to maintain.

At this time expert systems and Artificial Intelligence were beginning to be accepted into industry, so Ford contracted Inference Corp. to develop a prototype of the DLMS system. A rule-base approach was also considered, but the complexity and future maintainability of a system containing explicit knowledge about a dynamic domain such as automobile assembly ruled this approach out. A requirement for the DLMS knowledge base included the ability to make frequent and complex changes without affecting other components of the knowledge base. This required that the objects in the taxonomy be stored in classes that were analogous to the real world of automobile assembly planning. This approach led to a semantic network representation of the automobile assembly world where classes and subclasses corresponded to their appropriate equivalents in the real world. This type of semantic network representation was very similar to the KL-ONE representation language. It was decided to model the Ford automobile manufacturing knowledge base utilizing KL-ONE in order test the feasibility of this approach. This prototype proved very successful and the basic KL-ONE model proved to be

both robust and flexible as the knowledge base evolved over the years. Changes were made for processing and memory efficiency (i.e. the use of a hash table to store the list of concepts), but the KL-ONE logical design has been successful in terms of our problem domain.

DLMS KNOWLEDGE BASE STRUCTURE

As mentioned previously, the DLMS taxonomy or knowledge base contains all of the relevant information that describes the vehicle assembly process at Ford Motor Company. This includes all of the lexical classes included in Standard Language such as verbs, nouns, prepositions, conjunctions and other parts of speech, various tools and parts utilized at the assembly plants, and descriptions of operations that are performed to build the vehicle. Currently the DLMS taxonomy contains over 9000 such concepts.

The organization of the knowledge base is based on the KL-ONE model. The root of the semantic network is a concept known as THING which encompasses everything within the DLMS world. The children of the root concept describe various major classes of knowledge and include such things as TOOLS, PARTS and OPERATIONS. Each concept contains attributes or slots that describe that object. The values of these attributes are inherited from the concept's parents. Ranges of valid values can be given for any particular attribute. Any attempt to put an invalid value in that attribute will trigger an error. All of the information dealing with the organization and structure of the taxonomy is also contained in the taxonomy itself. There are four types of links that describe the relationship between any two concepts: subsumes, specializes, immediately-subsumes and immediately-specializes.

The subsumption relation describes a link between a parent concept and all of its children, including descendants of its children. The "immediately-subsumes" relation describes only the concepts that are direct children of the parent concept. The "specializes" and "immediately specializes" relations are inverses of the subsumption relation. A concept "immediately specializes" its direct parent concepts and "specializes" all of the concepts that are ancestors of its parents. These relationships are stored as attributes of any given concept and can be utilized as a tool to trace any concept through the entire taxonomy. Figure 4 shows how the DLMS taxonomy is organized.

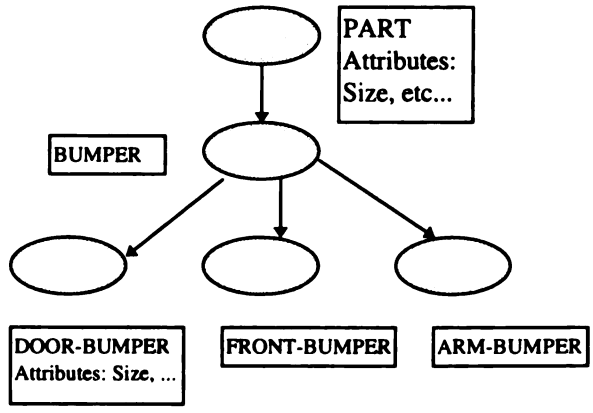


FIGURE 4: A Sample from the DLMS Taxonomy

KNOWLEDGE BASE MAINTENANCE

The DLMS Knowledge Base is maintained through the use of two different tools: the Knowledge Base Manager (KBM) and the Knowledge Base Update facility (KBU). The Knowledge Base Manager is a graphical tool that is used by the system developers to make important changes to the knowledge base that will affect the actual output generated by the system. Since this output will have a major impact on the assembly process any such change must be approved by a committee representing all of the interested parties. All changes made to the knowledge base are logged by the system to keep a record of the system's modification history.

The Knowledge Base Update (KBU) facility is used by system users to make minor modifications to the knowledge base. A minor modification is a change that will not impact the output produced by the system. Examples of minor modifications include the addition of new words into the taxonomy. The KBU facility allows users to incorporate these changes directly into the taxonomy without any kind of system developer intervention. All changes made through the KBU facility are also logged for future reference.

KNOWLEDGE BASE VALIDATION AND VERIFICATION

The DLMS system has been in production for over five years. In the highly competitive automobile industry, five years is a relatively long time as the cycle for bringing new models to market is steadily decreasing. Therefore, the DLMS knowledge base is being constantly updated and modified. It was necessary to develop tools that would prevent errors from being introduced into the DLMS knowledge base when making modifications to it.

One such tool is the use of an automated regression testing tool that runs a suite of test cases against the knowledge base. The results of the test cases are compared against a baseline file and all discrepancies are then flagged. These discrepancies are then examined manually in order to find out if this change is correct. The suite of test cases is constantly being updated to incorporate as many different test scenarios as possible.

Another approach that we are taking toward knowledge base validation is the use of automated tools that analyze the taxonomy. This analysis is based on knowledge of the problem domain as well as information about the structure of the DLMS taxonomy. For example, one requirement of the DLMS system is that all physical objects must have an attribute that describes their size. A scan of the taxonomy can then search for concepts in the class of objects that do not have a valid size. All concepts that may have an error are then flagged for future investigation by the systems organization. These types of utilities are very useful in finding errors which may contribute to decreased system accuracy. Our future plans for DLMS include moving a large portion of the knowledge base maintenance activity to the user community, which will not have a technical background. To accomplish this we are also developing various automated knowledge base verification and validation routines that will point out errors as they are being introduced into the knowledge base.

This utility is based on the concept of a "knowledge base metric", that is roughly analogous to software metrics that are used as a quantitative measure of software quality. Software metrics are computed by measuring various attributes of code in order to develop a value that can be used to predict the complexity and maintainability of this program [Khoshgoftaar, Oman 94]. Our goal is to develop a knowledge base metric that will be computed automatically when the knowledge base is modified. This number is computed by counting the number of discrepancies that have been found by applying the specific DLMS utilities to the knowledge base and using that number as a baseline. After each modification this number is recomputed and then compared to the baseline. If the new metric is greater than the baseline this change will be flagged as it may have introduced additional errors in the knowledge base. This metric will provide us with an additional tool of verifying the knowledge base correctness.

CLASSIFICATION IN DLMS

The DLMS system utilizes a classification algorithm to create concepts and place them into their appropriate position in the taxonomy. The classifier utilizes various attributes of the concept in order to place it into its correct position. These "classifiable" attributes are slot values that play a major role in determining where this concept belongs. For example, the attribute "size" is very important in classification, while the "output format" slot has little value in classification. This classification is performed by finding the appropriate subsumers, linking the concept in and then locating all the concepts that should be subsumed by the new concept. The system narrows this search procedure considerably by selecting the appropriate node in the concept to begin the classification process. The concept which is to be classified is placed at the starting node; the system then tries to push the new concept node as far down the tree as possible. The classifiable attributes are used as a objective measure to determine if the concept is in its proper place. Within DLMS this classification algorithm is applied to all of the instances of the input string that describe the process element. In a simple element this may include a verb, an object and an associated tool. When the classifier is complete, each of the above instances will inherit necessary values from the knowledge base in order to build the appropriate operation to describe required actions. For a more complete explanation of the DLMS classifier see [Rychtyckyj 94]. Figure 5 illustrates a diagram of a simple classification procedure.

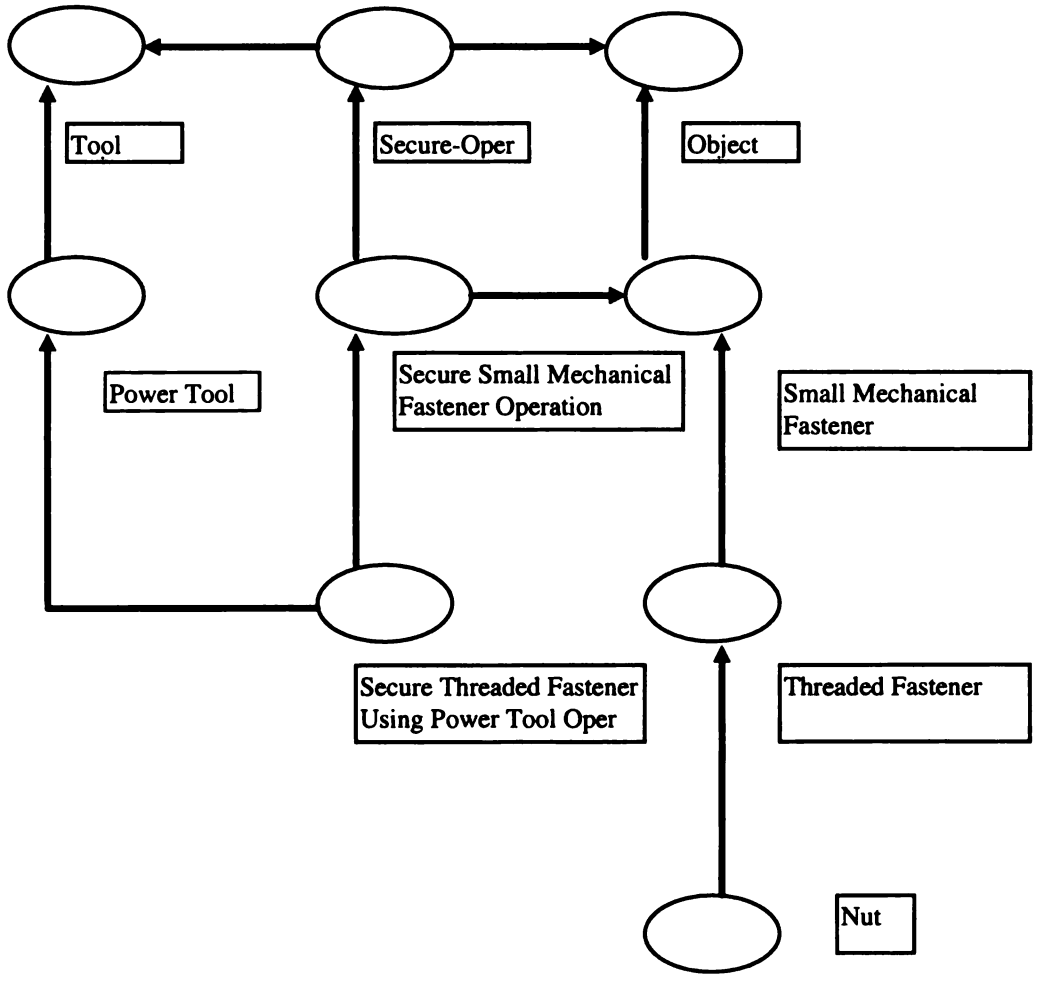


Figure 5. Taxonomy Classification Structure for Secure Operation

This figure describes the classification procedure for an operation of type "Secure Nut Using Power Tool". The concept "Nut" belongs to the class of "Threaded Fasteners". The analyzer subsystem also finds the class of Power Tools and then attempts to find which operation most closely matches the derived objects. In this case, the classifier has identified the power tool, the "Secure" verb and the object "nut". It will now try to find the "Secure Operation that most closely matches these parameters. In this case the Secure Threaded Fastener Using A Power Tool is the closest match and the classifier will then use this operation to start building its output solution.

EVALUATION OF KL-ONE IN DLMS

This section will describe the advantages and disadvantages of the KL-ONE representation scheme as it has been utilized within DLMS. Along with Classic [Brachman, et al. 91] DLMS has been one of the few KL-ONE-based systems that has been utilized in production for a considerable amount of time. KL-ONE was selected as the appropriate tool for this project because of the requirement for a flexible and powerful knowledge management tool. The knowledge base required to store information about the vehicle assembly process is both complex and dynamic. Over the last two years Ford Motor Company has replaced or changed more than half of its product line in North America, including its two top selling vehicles: the Taurus and F-Series pickup truck. Such a rate of

change requires that the DLMS system be able to incorporate continuous updates to its knowledge base. These changes also included major modifications to the structure of the taxonomy, as well as those related to maintaining the actual automobile assembly knowledge itself.

The most complex part of the DLMS taxonomy is the class of concept known as an OPERATION. An operation is a concept that describes all of the individual actions that must be done at the assembly plant. A simple operation, such as "Hammer Object" contain the following slot values: Tool (tool required), Modapts (Modapts codes that describe this operation and an Actions slot. In this case the Actions slot will be empty, because no additional actions are required. However, the vast majority of operations require a multitude of actions, so this value will contain a list of the other required actions. The system will then process each of these actions in turn, and it is quite likely that the secondary actions may also contain subordinate actions. Therefore, the output generated from a single operation may be accessed from many different places in the taxonomy. This flexibility makes it possible to model the real world in DLMS, but it also makes it very difficult to trace where an error may have occurred. This is very analogous to tracing the execution of a large rulebase to find a single rule that has incorrectly fired. It is also possible to introduce cycles into the taxonomy; an indirect cycle of the form A -> B -> C -> A will usually not be noticed until an actual run goes into an infinite loop. This tradeoff between flexibility and ease of maintenance is common as the complexity of the world we are trying to model increases. As described previously in the section on knowledge base validation and verification, our approach is to build more sophisticated tools that incorporate intelligence about both KL-ONE and the automobile assembly process in order to improve the overall maintenance.

Another problem we have encountered is the requirement from various user organizations to produce documents describing the various taxonomy outputs. The complexity of the taxonomy makes any general reports extremely long and difficult to follow. Our approach has been to educate the user community about the structure of the DLMS taxonomy in contrast to a spreadsheet or a database, which are well understood by most people. Another issue that we must be careful with is the fact that taxonomy modifications may change the output from a previous run. The user community is not very receptive when a process sheet that worked last year is now flagged as invalid, because we have improved the error-checking capability of the system.

To forestall most of these problems we disseminate any such changes to the user community when they are implemented in order to prevent any future misunderstandings. Overall, based on our years of experience with the KL-ONE representation system, we are quite satisfied that KL-ONE has proved to be a very capable tool and has contributed greatly to the success of the DLMS system.

CONCLUSIONS

In this paper we have tried to describe our experience at Ford Motor Company with the KL-ONE representation system. The DLMS system is a production system that is utilized by hundreds of users throughout the company. Therefore, the reliability of the system is of the highest priority and any new technology must be robust enough to operate successfully in this environment. We have discussed the ongoing problems of knowledge base maintenance and knowledge base validation and verification. Our solutions to both problems include automated tools and manual intervention by the system developers as required. Our goal includes the development of more sophisticated tools that would eventually pass all knowledge base maintenance activity to the user community. We have also discussed the ramifications of using a complex knowledge representation system in the realm of a corporate business environment and the steps we have taken to improve this situation. Our extensive use of KL-ONE within DLMS has convinced us that the KL-ONE representation system is an excellent tool for modeling the complex world of vehicle assembly planning and we are planning to expand our usage of KL-ONE into other problem domains in the automobile industry.

Acknowledgments

The Direct Labor Management System was a product of the work of many people that have contributed to the success of the system over the years; therefore I would like to give credit to the following people: John O'Brien, Tom Kaszamarek, Scott Hatfield, Wayne Johnson, Richard Woodhead, Alan Turski, Henry Brice, Tom Vitale, Jay Zaback and Rick Keller.

I would also like to thank Dr. Robert Reynolds of Wayne State University for all of his valuable insights and suggestions. I am also indebted to Alan Turski, whose work on the DLMS system was also instrumental for its success. Thanks are also due to the anonymous referees, whose comments and suggestions I have tried to incorporate into this paper.

References

Baade, F., (1990), "Terminological Cycles in KL-ONE-based Knowledge Representation Languages" in Proceedings of the Eighth National Conference on Artificial Intelligence, vol. 2., pp. 621-626.

Brachman, R., McGuiness, D., Patel-Schneider, P., Resnick, L., Borgida, A., (1991) "Living With Classic: When and How to Use a KL-ONE-Like Language" in Principles of Semantic Networks, ed.
J. Sowa, pp. 401-456, Morgan Kaufmann Publishers.

Brachman, R., Schmolze, J.(1985), "An Overview of the KL-ONE Knowledge Representation System", Cognitive Science 9(2), pp. 171-216.

Charniak, E., Riesbeck, C., McDermott, D., Meehan, J., (1987), Artificial Intelligence Programming, pp. 304-336, Lawrence Erlbaum Publishers.

Industrial Engineering Services (1988), Modapts Study Notes for Certified Practitioner Training.

Khoshgoftaar, T., Oman, P., (1994), "Metrics in Software", IEEE Computer, Vol. 27, No. 9, pp. 13-15.

Lipkis, T., "A KL-ONE Classifier", (1981), Consul Note #5, USC/Information Sciences Institute.

O'Brien, J., Brice, H., Hatfield, S., Johnson, W., Woodhead, R. (1989), "The Ford Motor Company Direct Labor Management System: in Innovative Applications of Artificial Intelligence, ed. Schorr & Rappaport, pp. 331-346, MIT Press.

Rychtyckyj, N. (1994), "Classification in DLMS Utilizing a KL-ONE Representation Language" in Proceedings of the Sixth International Conference on Tools With Artificial Intelligence, pg. 339-345, IEEE Computer Society Press.

Schmolze,J., Lipkis, T.,(1983), "Classification in the KL-ONE Knowledge Representation System" in Proceedings of the Eighth International Joint Conference on Artificial Intelligence, pp. 330-332, Morgan Kaufmann Publishers.

Woods, W., Schmolze, J.,(1992), "The KL-ONE Family" in Computers & Mathematics with Applications, Vol. 23, No. 2-5, pp. 133-177.

On Chronicles : Representation, On-line Recognition and Learning

Malik Ghallab
LAAS-CNRS, 7 Avenue du Colonel Roche
31077 Toulouse, France (malik@laas.fr)

Abstract

A knowledge representation for chronicles is proposed. It enables to model temporal scenarios whose occurrences need to be recognized on-line, e.g., for supervision applications. It relies on domain attributes reified by predicates *event* and *hold* for expressing change and persistence, and on temporal constraints. It has been chosen such as to keep the algorithmic complexity of the recognition system reasonable while maintaining expressiveness. It has been field tested in several applications. The paper describes the representation, the on-line recognition process, and a learning-based approach for synthesizing chronicle models from samples.

1 INTRODUCTION

The supervision of a complex dynamic process requires the surveillance, through sensing devices and signal processing systems, of specific events. But it also requires a reasoning mechanism, above the detection system, for recognizing particular evolution described as set of temporally related events. This is the kind of chronicles we are interested in here. A chronicle model is a description of generic scenario of a normal or an abnormal evolution to be surveyed. It is represented as a set of events and a set of temporal constraints, between these events and with respect to a context. A chronicle model may also specify events to be generated and actions to be triggered as a result of the chronicle occurrence. Deduced events can in turn be taken as input by other chronicles, hence enabling a recursive chaining.

On-line, the chronicle recognition system receives as input a stream of time-stamped instanciated events. A sensory stimulus, once detected by a signal processing tool, may become an event. If some observed events match the model events of a chronicle, and if their occurrence dates meet the specified constraints, then an instance of this chronicle occurs. Its recognition may generate in turn new events, it can permit the focus of attention enabling the detection of forthcoming events, or it can trigger alarms, messages, data logging or other actions. This recognition system recognizes all instances of occurring chronicles, on the fly, as they develop. It does not account for all observed events, but only for those matching available models. The system generates as output deduced events and triggered actions associated to recognized chronicle models. It is mainly a temporal reasoning system. It is predictive since it predicts forthcoming events that are relevant to partial instances of chronicles currently hypothesized as taking place; it focuses on them and it maintains their temporal windows of relevance.

Several areas of research investigate problems close to ours. In plan recognition for example, one is interested in recognizing that a sequence of observed actions matches a previously described plan. In that area, a top-down approach similar to ours where recognition is guided by the plan to be recognized, is proposed in [Rao 94]; in addition to several restrictive assumptions (mainly the synchronization between observations and action occurrences) this work, as most others in plan recognition, does not take into account explicit temporal and real-time constraints. Time is explicit in the event calculus of Kowalski and Sergot. This representation, mainly developed as a question-answering system for temporal data-bases [Kowalski 92], which has been extended for planning or scheduling by several authors (e.g. [Borillo et al 91]), does not address the recognition issue. A variant of it based on the interval calculus [Kumar 87] is closer to our problem, but with a more restricted representation and without algorithmic concerns. In diagnosis, several authors consider dynamic systems; among which [Console 92] and [Nejdl 94] explore available knowledge on temporal constraints between states (called modes) mainly as a way to focus an abductive diagnosis. On-line recognition is not their main focus, they do not provide complexity analysis, nor performance results.

In contrast, our knowledge representation has been chosen such as to keep the complexity reasonable for on-line processing, while maintaining a sufficient expressiveness for practical applications. The algorithms developed rely

on extended pre-processing and carefully chosen data structures. The corresponding system, called IxTeT, has been field-tested on industrial applications, mainly for the supervision of gas turbines. IxTeT exhibits good performances, compatible with real-time requirements.

This paper summarizes the proposed knowledge representation. It gives an outline of the chronicle recognition approach developed and performance achieved so far. It describes a learning-based approach which synthesizes chronicle models from positive and negative examples. It discusses in conclusion the limitations and advantages of the proposed contribution.

2 KNOWLEDGE REPRESENTATION FOR CHRONICLES

Our knowledge representation relies on temporally qualified domain attributes. We consider time as linearly ordered discrete set of instants, whose resolution is sufficient for the dynamics of the environment (i.e., any change can be adequately represented as taking place at some instant of the set). For algorithmic complexity reasons our time-map manager uses time-points as the elementary primitives. Intervals and relations of the restricted interval algebra [Vilain86, Nebel93] can also be represented at the user level in IxTeT, but they are translated internally into time-point constraints (we'll not develop that issue here). We can handle the usual symbolic constraints of the time-point algebra (i.e. *before*, *equal*, *after* and their disjunctions), as well as numerical constraints. The later are expressed as pairs of real numbers *(e2-e1)=[I-, I+]* corresponding to lower bounds and upper bounds on the temporal distance between two points e1 and e2. For complexity reasons, we do not allow disjunctions of numerical constraints, since the constraint satisfaction problem becomes NP-Complete [Dechter91].

We rely on a propositionnal reified logic formalism [Shoham87], where a set of multi-valued domain attributes (fluents) are temporally qualified by two predicates: *hold* and *event*. The persistence of the value v of a domain attribute p during the interval $[t, t'[$ is expressed by the assertion *hold(p:v, (t,t'))*.

An event is an instantaneous change of the value of a domain attribute, the predicate *event* is defined through the predicate *hold*:

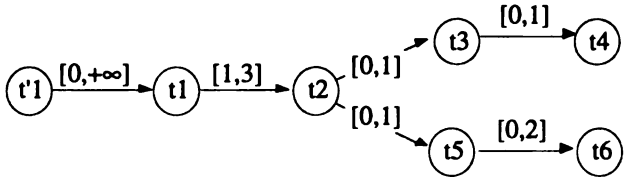
$$event(p:(v_1,v_2),t) \underset{def}{\equiv} \exists \tau < t < \tau' \mid hold(p:v_1,(\tau,t)) \wedge hold(p:v_2,(t,\tau')) \wedge (v_1 \neq v_2)$$

The representation allows multi-valued attributes with variable arguments, each instance of the arguments corresponds to an independent fluent (e.g. p(?x) gives p(a), p(b)...). However the representation does not allow free variables in detected instances of events (semi-unification only between model events and observed event instances) nor in recognized instances of chronicles.

A chronicle model is a conjunction of *event* predicates and constraints. It may also involve the description of some context that is required to hold over some interval independently of when the corresponding fluents became true. For example, we want to survey the correct reaction of a turbine to a load increase (fluent *LOAD*). The response must be an increase of the shaft speed (*SHAFT*) followed by an opening of the fuel valve (*VALVE*) and an increase of the mean exhaust temperature (*EXHAUST*). We must also observe an increase then a decrease of the angle of the nozzles coupling the high pressure and the low pressure parts (*NOZZLE*). During all this chronicle, the turbine needs to be in a particular state at least until the *VALVE* opens (state represented by the fluent *ALARM* set to *NONE*). This chronicle model is described as follows:

```
chronicle IncreasingLoad
{       timepoint t1,t2,t3,t4,t5,t6;
 //- forthcoming events and context assertion
        event (LOAD:(Steady, Increasing),t1);
        event (SHAFT:(Steady, Increasing),t2);
        event (VALVE:(Steady, Increasing),t3);
        event (EXHAUST:(Steady, Increasing),t4);
        event (NOZZLE:(Steady, Increasing),t5);
        event (NOZZLE:(Increasing, Decreasing),t6);
        hold (ALARM:None, (t1, t4));
 //- temporal constraints
        (t2-t1) in [1.00,3.00];
        (t3-t2) in [0.00,1.00];    (t4-t3) in [0.00,1.00];
        (t5-t2) in [0.00,1.00];    (t6-t5) in [0.00,2.00];
  when recognized
        report "Successful increasing load";    }
```

The system receives and processes only events. The chronicle recognition is complete as long as the observed event stream is complete, i.e. any change of the value of a domain attribute produces an observed event. This hypothesis enables to manage context assertions quite naturally through occurrences and non-occurrences of events. To process assertion *hold(p:v, (t,t'))*, the system checks that there has been an *event (p:(v',v),t")* with $t"<t$ and such that no event *p:(v,v")* occurs within *[t",t'[*. To initialize the system, a set of events corresponding to the initial state of the world must be sent with an occurrence date equals to $-\infty$. Consequently a chronicle model corresponds to a network of temporal constraints where each node is an event. The network for the previous chronicle is the following:



with the condition that no event *ALARM:(NONE,?t)* occurs during *[t'1,t4[*.

The user can also specify some actions to be performed during or after a chronicle recognition. These may generate a new event with occurrence date relative to those of the instance events, may display messages, or run a user specified procedure (external action). The generation of a new event allows to relate different chronicles. An action can be linked to a complete recognition of a chronicle instance, or to a partial recognition after the occurrence of some event. This provides a *focus of attention* capability.

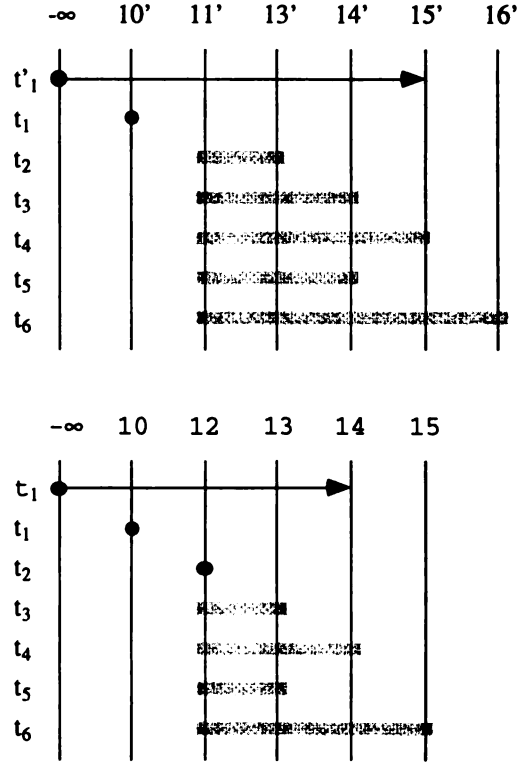
3 ON-LINE CHRONICLE RECOGNITION

The chronicle recognition system has to detect, on the fly, any subset of the stream of observed events which matches the set of model events in a chronicle and meets all the constraints and context assertions. A match between some observed events with a subset of model events is a partial instance of a chronicle. A complete match requires the observation of all model events. However, not all observed events have to be accounted for: an event is meaningless unless it happens within a context and a sequence of events matching a chronicle

Let us illustrate the recognition process with a simple scenario corresponding to the chronicle model given above: initially *ALARM* is set to *None*, the *LOAD* increases at time 10', the *SHAFT* speeds up at 12' and so on. On receiving a *LOAD* increase event at 10', the system detects a possible instance of this chronicle, it predicts what are the expected events for this instance and what should be their respective temporal windows compatible with this partial instance (top figure on the right).

Reception of an event on *SHAFT* speed up at 12' is propagated to the windows constraining the remaining events (bottom figure). If, on some alternate scenario, after receiving these two events, either *VALVE* or *NOZZLE* events do not occur at time 13', or if *ALARM* changes its value before occurrence time of *EXHAUST* event, then the corresponding instance will not meet the specified constraints and will be disregarded.

A compilation stage is useful for testing the consistency of constraints in a chronicle model and for coding it into data structures efficient for the recognition process. It mainly consists into propagating all the constraints into a complete network. At preprocessing time, propagations are local to each chronicle model. For each chronicle model, we propagate constraints with an incremental path consistency algorithm (derived from [Mackworth85]). The result of this algorithm is (for each chronicle model) the least constrained complete graph equivalent to the user constraints, or a subset of inconsistent constraints if any.



On-line, an event *e* is observed with a variable delay, due to sensor processing and data transmission. We suppose that for each domain attribute this delay for corresponding events is upper and lower-bounded within an interval $\Delta(e)$ given by the user. If $d(e)$ is the occurrence date and $r(e)$ the reception date of event *e*, we always have: $r(e) \in \Delta(e) + d(e)$.

An observed event may trigger a hypothesis of a chronicle instance. Such a partial instance corresponds to a network, grounded in time, predicting expected events for this hypothesis. Events that may trigger a recognition hypothesis are those at the root nodes of the model network as well as at any further nodes compatible with the bounds on detection delays Δ of the predecessor nodes and the constraints.

The recognition process relies on a complete forecasting of expected events predicted by chronicle hypothesis. We define an interval, called *window of relevance D(e)*, which

contains all possible occurrence dates for a predicted event e of a partial instance S, in order for e to be consistent with constraints and known dates of observed events in S. Furthermore, for each chronicle instance, we maintain some time bounds. The *deadline* of an instance S is the latest date for the nearest non-occurred event. Similarly, we compute the *holdline* of an instance which is the latest date for ensuring that the nearest assertion holds, i.e. when we are sure that an occurrence of a forbidden event cannot violate the assertion (for the previous example, at time 10' *deadline*=16' and a *holdline*=15'). We can compute the *recognition interval* for each chronicle instance: the user can know when a recognition is expected and which events will be necessary to achieve it. This is used for performing some actions such as a focus of attention on what is expected or what needs to be avoided.

A hypothesis of chronicle instance may progress in two ways: (i) a new event may be detected, it can be either integrated into the instance and make the remaining predictions more precises, or it may violate a constraint for an assertion and make the corresponding hypothesis invalid; or (ii) time passes without nothing happening and, perhaps, may make some deadline violated or some assertions constraints obsolete. The system manages its own internal clock by receiving clock updates. Let *now* denoted the value of this clock. When an observed event e matches of model event e_k, we always have the reception date $r(e) = now$, and either

- $d(e) \notin D(e_k)$: e doesn't meet the temporal constraints of the expected event e_k of S,

- $d(e) \in D(e_k)$: $D(e_k)$ is reduced to the single point $d(e)$.

More generally, if the window of relevance of some expected event e_k in S has been further constrained (that is $D(e_k)$ reduced), we are sure that this constraint remains consistent with what is already known. We need however to propagate the reduction to other expected events, which in turn are further constrained.

propagate(e_k, S)

for all forthcoming event $e_i \neq e_k$ of S

$$D(e_i) \leftarrow D(e_i) \cap [D(e_k) + I(e_i - e_k)]$$

This produces a new set of non-empty and consistent $D(e_i)$. We never need to verify the consistency of events that fall into their windows of relevance. We just need to propagate further their new value of $D(e)$.

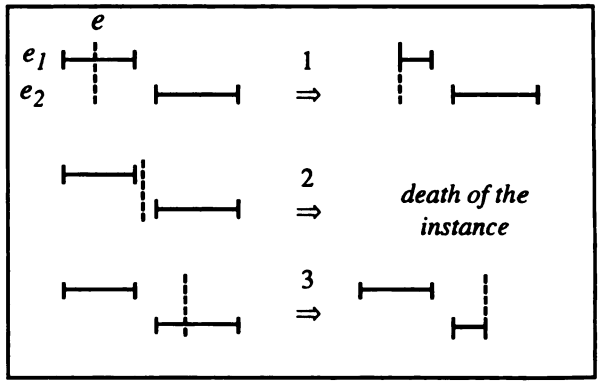
When the system updates its internal clock, this new value of *now* can reduce some windows of relevance $D(e_i)$ and, in this case, we need to propagate it over all expected events of an instance S (by $D(e_i) \leftarrow D(e_i) \cap ([t, +\infty[- D(e_i))$). This instance remains consistent as long as all $D(e_i)$ remain non empty.

Notice that a clock update may not require propagation: it is necessary to propagate a date only when a time bound is reached (before that we are sure that constraints in a chronicle instance remain consistent). Time bounds enable an efficient pruning.

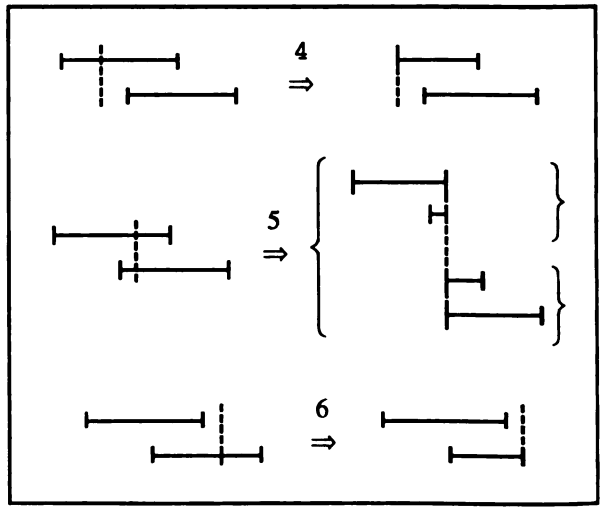
Checking *hold* assertions requires a special care. Let us consider a chronicle instance S which requires that an event e shouldn't occur during [e1,e2]. If e occurs before D(e1) or after D(e2), S isn't concerned. Otherwise, there are two basic cases (depending on the overlap between D(e1) and D(e2)) and each one leads to a processing according to the localization of the occurrence date of e.

In four cases out of six (illustrated below), we only reduce D(e1) or D(e2), but S can also die (case 2) or can be cut in two more restricted instances (case 5). We propagate these new dates by processing propagate(e,S) if D(e) is reduced and/or propagate(e',S) if D(e') is reduced.

$D(e_1) \cap D(e_2) = \emptyset$



$D(e_1) \cap D(e_2) \neq \emptyset$



Hypothesis management

Before propagating a second observation into a hypothesis, the original chronicle instance must be duplicated and only the copy is updated. For each chronicle model, the system manages a tree of hypotheses of current instances. When a hypothesis of chronicle instance is completed or killed (because of a violated constraint), it is removed from this tree. Duplication is needed to warranty the recognition of a chronicle as often as it may arise, even if its instances are temporally overlapping.

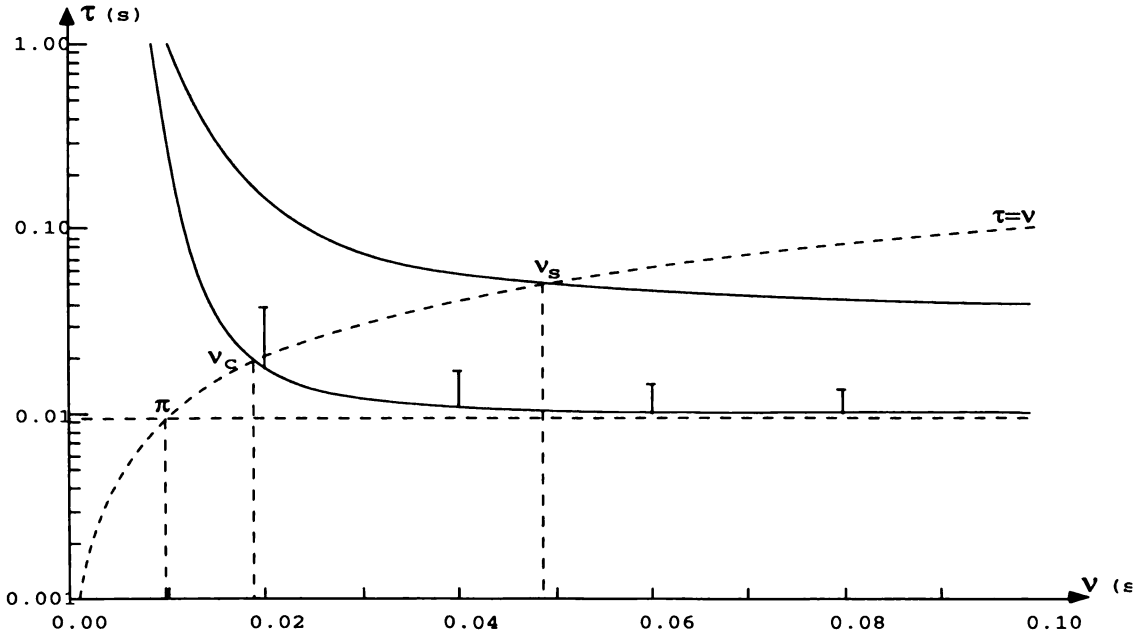
To illustrate this need consider for example a simplified chronicle model with 3 events where : e1 < e2 < e3 with (e3 - e2) ≤ 3'. If we receive the stream of events: e1 at time 0', e2 at 3', a second instance of e2 at 8', e3 at 11' , the first occurrence of e2 (if matched) will kill this chronicle instance hypothesis because of the non occurrence of e3 before time 6'. By keeping a second instance of the chronicle (when receiving the first e2), we can recognize the instance matching the second instance of e2 at 8'. The size of the tree of hypotheses is the main source of complexity. Duration bounds in a chronicle model is first way to reduce it. Further knowledge restricting multiple instances of events is also beneficial. There may also be chronicles that cannot have two complete instances which overlap in time or share a common event; the user may also be interested in recognizing just one instance at a time. For both cases, when a chronicle instance is recognized, all its pending instances must be removed, this can be done by using an assertion overall a chronicle.

Applications and Performance results

The IxTeT recognition system has been extensively tested, on laboratory data as well as on industrial applications. One of these was concerned with the supervision of gas turbines. It took place in the context of Esprit project TIGER, which demonstrated its results on a large turbine powering a petrochemical plant, as well as on an aircraft APU turbine. The corresponding supervision system is currently marketed [Milne 94]. The proposed representation has also been integrated to a system for the supervision of a blast furnace. IxTeT is currently being tested for the supervision of the electrical power net.

The algorithms developed here have a polynomial time complexity: each elementary event propagation or clock propagation runs in $O(m^2)$, where m is the number of pending events in a chronicle instance; $m \leq n$ the total number of distinct events in a chronicle model. For K instances of chronicle models with n events each, the proposed system processes incrementally a new event (or a clock update) with a complexity in $O(Kn^2)$. K is larger than M, the number of chronicle models. If K is kept of the same order than M, which is the case if the duration of all chronicle models is upper-bounded, the overall complexity is quite manageable.

A large number of experiments were run on different sets of chronicles (up to a hundred chronicle models). One experiment aimed at characterizing the practical "bandwidth" of the proposed algorithms and implementation: up to which frequency of events the system is able to follow-up without accumulating delays.



For a given set of chronicle models, we systematically generated scenarios of random but meaningful sequences of events (i.e., leading to recognized chronicles). Each scenario corresponds to some average event frequency. Temporal constraints where parametrized with respect to this frequency such as to keep chronicle models in accordance with the test scenario, and to maintain the number of of recognized chronicles significantly large and independent of the event frequency.

The preceding chart summarizes the system performances (measured on a Sparc II station) for a set of 80 chronicle models, each involving about 10 events and as much as constraints, 2 or 3 of which referring to context assertions. For each event frequency, 100 scenarios involving about 3000 events each where ran. For each scenario, about 150 chronicle instances were recognized.

The figure plots the response time τ with respect to ν the average period between 2 events ($1/\nu$ is the event frequency). π is the average processing time of an event, it does not depend on ν. The response time τ is the difference between the reception date r(e) and the time at which event e has been completely processed and the system starts processing the next waiting event. Hence τ takes into account the delay in a wait buffer from reception time to the beginning of processing time: $\tau \geq \pi$. The figure shows the maximum values and the average values (with few standard deviations) of τ. Three interesting points with respect to the diagonal curve $\tau = \nu$ define four bands

- $\nu_S \leq \nu$: a *secure band* where there is never a delay,
- $\nu_C \leq \nu \leq \nu_S$: a *safe band*, there may be a delay from one event to the next which does not grow since the average response time is smaller than the average period,
- $\pi \leq \nu \leq \nu_C$: a *critical band* in which the system may or may not catch up after some sequence of events, and
- $\nu \leq \pi$: a *forbidden band* where cumulated delay grows with the number of events; the system cannot follow up.

Similar curves were obtained for other sets of chronicle models. The processing time π grows clearly with the number of chronicles, with values π = 6, 8, and 9 ms for 20, 50 and 80 models respectively. A similar remark holds for ν_C (ν_C = 16 and 20 ms for 20 and 80 models).

4 LEARNING CHRONICLES

Expressing a domain knowledge as a set of chronicle models is not a trivial task. To ease down the modeling efforts, we explored a learning-based approach. Its purpose is to generate from a set of recorded examples of a given behavior a chronicle model which recognizes each instance of this sample set, but does not recognize any sample from counter-examples, i.e., other recorded behaviors. The method described below, inspired from learning techniques used in syntactical pattern recognition and now explored in biogenetics, synthesizes automatically discriminative *skeleton models* containing only events and constraints, without context information (i.e., no *hold* predicate). These skeleton models have to be completed by the programmer with context information and eventually with some tuning of the bounds of numerical constraints.

Typically, for handcoding a chronicle model, the programmer analyses a set of charts recording raw sensor signals focused over periods exhibiting the behavior to be modeled, e.g. periods preceding a fault or a transition between control modes. Not all available sensory signals appear in such a charts, but only those possibly relevant for the behavior to be modeled. Simple patterns are extracted from these signals, such as a significant change in the value of the signal or of its derivative with respect to a threshold. More complex patterns, e.g. oscillations or wavelets may be used if needed. For our purpose, a recorded example for a particular behavior is a set of time stamped patterns on raw signals which constitutes a sequence of event instances. Typically for a dozen of behaviors, there are about 10 to 50 recorded examples for each one, an example is sequence containing 20 events from a total of about 100 event types.

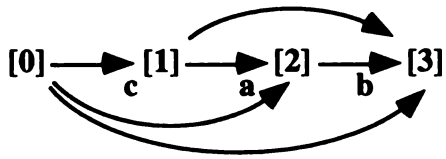
The learning method developed takes as input a collection of sequences of events, each sequence labeled by the corresponding behavior. The method gets rid momentarily of temporal information and reduces these sequences of events to strings over some alphabet. If we assume that this set of strings can be characterized as a regular language, efficient polynomial algorithms are known which enable to synthesize from positive and negative examples a finite automata which recognizes this language (see [Oncina 92]). However this assumption seems not to fit our representation. We preferred to rely instead on a more flexible structure: automata of subsequences of a string and their Cartesian product [Hebrard 84]. Our apporach clusters a set of strings with respect to an adequate measure. To each cluster is associated an automaton of maximal common subsequences. A conjunct of maximal subsequences, discriminative with respect to counter examples is extracted from this automaton. This provides the skeleton of a chronicle model, to which temporal bounds from recorded sequences are further added. The result is interactively checked out and augmented by the user.

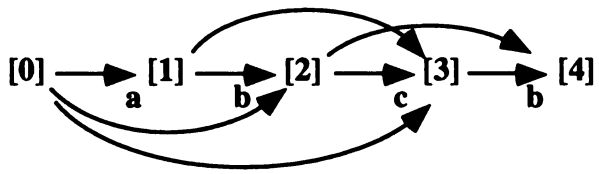
More formally let S, S', S", ... be a collection of samples; each sample $S = \{s_1, s_2, ..., s_m\}$ is a set of strings over an alphabet Σ (i.e., propositionnal letters denoting possible event types). Sample S corresponds to a behavior $\lambda(S)$; the

strings in S are recorded examples of this behavior. With respect to $\lambda(S)$, all strings in the other samples S', S", ... are counter-examples.

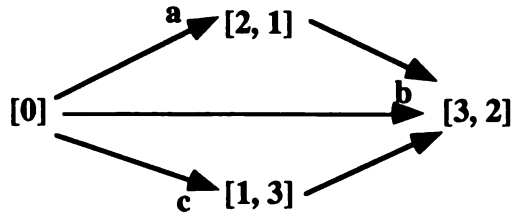
We note a string s= s[1]s[2]...s[p]. A ***subsequence*** of s is a string σ of length k, $k \leq p$, such that there are k indexes $i_1 < i_2 < \ldots < i_k$ with : $\sigma[1]=s[i_1]$, $\sigma[2]=s[i_2]$, ... and $\sigma[k]=s[i_k]$. For example "choice" is a subsequence of string "chronicle" (but not a substring of it).

Let A[s] be the automaton of subsequences of s. A[s] is easily defined. It has (p+1) states: the initial state and one state for each letter in s. Transitions from state [i] are labeled by all distinct letters in s following letter s[i]. A transition labeled by the letter s[j] leads to state [j] if j is the first occurrence of this letter following s[i], we note $\delta([i], s[j])=[j]$. The figure below shows the automata of subsequences for strings "cab" and "abcb". Note that all arcs leading to [j] are labeled by the same letter, and that A[s] is acyclic. While checking some input string, if automaton A[s] is in state [i] and if it reads a letter α which does not label any arc outgoing from [i], then this input is rejected. Otherwise one proceeds to state $\delta([i], \alpha)$ until either the input is rejected or until the input is finished being read; in that case the input is accepted as a valid subsequence of s. A[s] recognizes all and only the subsequences of s. One may use a straightforward algorithm for building A[s] in time complexity $O(p.\ \min\{p, |\Sigma|\})$.





The Cartesien product of 2 automata A[s]xA[s'] is an automaton with an interesting property: it recognizes all and only the subsequences common to the 2 string s and s'. A state in this automaton is denoted by the product [i, j]. If [i] and [j] are states in respectively A[s] and A[s'] with transitions $\delta([i], \alpha)$ and $\delta([j], \alpha)$ labeled by the same letter α then there is a transition in the product automata labeled by letter α: $\delta([i,j], \alpha) = [\delta([i], \alpha), \delta([j], \alpha)]$. Since [i] and [j] are not necessarily followed by identical letters, the product automata has usually much less states then the maximum number of 1+pp'. One may also merge two equivalent states (i.e., those with the same label on their incoming arcs and with identical outgoing arcs) in the product automaton to reduce it. The result is still a deterministic acyclic automaton. The following figure shows the Cartesien product of the two automata given above (where states [3,2] and [3,4] have been merged).



The Cartesien product of m automata $A[s_1]x...xA[s_m]$, which can be computed incrementally and independently of the order of processing, provides the same nice property: it recognizes all and only the subsequences common to the m strings s_1, ..., s_m.

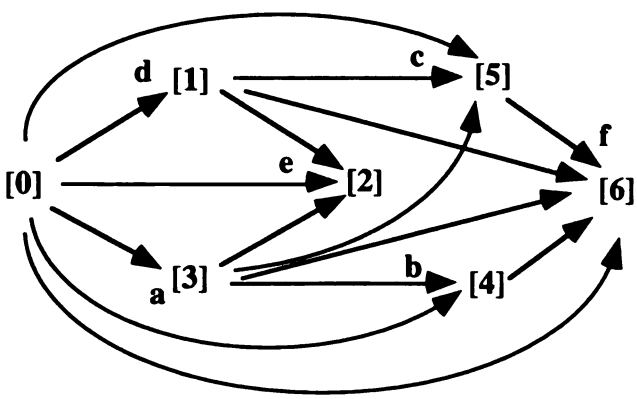
Let us note LCS(s, s') a longest common subsequence to s and s', and lcs(s, s') its length. An LCS corresponds simply to a longest path in the acyclic product automata A[s]xA[s']; there can be several LCS with the same length, e.g., "ab" and "cb" in the above automaton. The measure lcs provides a good distance between two strings: the larger is lcs(s, s') the closer the two strings are. This measure appears to be more robust than other well known distances on strings (e.g. the Hamming distance); it is closely linked to the editing distance [Hebrard84]. For our purpose, lcs also provides the common characteristic features to two strings. Note that lcs is a commutative and associative measure.

The idea is to characterize a set of strings by their longest common subsequences and to extract from these LCSs the squeleton of the chronicle model. The problem is that a sample $S= \{s_1, s_2, ..., s_m\}$ may not be homogeneous with respect to the lcs measure: their LCSs could too small or even empty and hence meaningless. We may need to cluster the sample S into homogeneous classes with respect to lcs.

This is just what we have done. We did not rely however on the technique of the Cartesien product automata for the clustering purpose, but on more specific and efficient algorithms. Indeed, for clustering S we just need to evaluate the measure lcs, but do not need to exhibit common subsequences. The algorithm of [Hunt 77] gives the exact value of lcs(s, s') in quadratic time; it is inspired from and improves significantly over a dynamic programming approach. We use it to compute the binary distance between all couples of strings in sample S. The algorithm of [Guenoche 92] gives a lower bound estimate of the $lcs(s_1, ..., s_k)$ for k strings; its complexity is in $O(|\Sigma|kl)$, where l is the value of the found estimate for lcs. This algorithm is quite efficient and provides meaningful

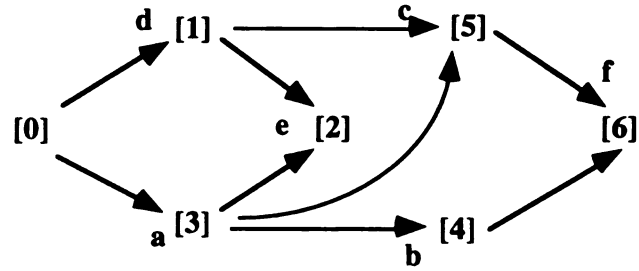
estimates (in the average, about 90% of the exact value for k=10 strings of length 100 characters each). We use this algorithm to augment a cluster of strings, starting from the closest pairs given by algorithm of [Hunt 77]. For the moment our clustering method is a heuristic compromise between over generalization (few large clusters with small lcs) and over specification (many small clusters with large lcs), with a bias towards the former case, since we are using counter-examples to choose among the LCSs of a cluster. Our heuristic criterion is mainly the difference lcs(s, s') - l(s, s', ..., s_k), l being the estimated longest length; i.e., we augment the cluster with s_k as long as this difference is below some threshold. At the end of this procedure, we compute for each cluster on strings in S the Cartesien product of the corresponding subsequence automata. Let A_C be this automaton.

The figure below gives the product automata A_C for the 3 strings "abdcfe", "adbcef" and "daecbf".



A_C provides not only all the LCSs of the strings in the cluster, but also an interesting superset of these LCSs : common subsequences *maximal* with respect to the subsequence partial order. A subsequence s is maximal if there is no other subsequence recognized by A_C of which s is a proper subsequence. Note that s can be maximal and not a longest one. Any subsequence common to the strings in the cluster is a subsequence of some maximal one. Hence maximal subsequences in A_C account for all subsequences common to the cluster. These maximal strings are directly obtained from A_C.

We can as easily transform A_C into an automaton A_M which recognizes all and only maximal subsequences: we just need to remove from A_C any arc between the two states which are linked by another longer path. The corresponding automaton A_M is displayed hereafter (the maximal subsequences are "ae", "de", "abf", "acf" and "dcf").



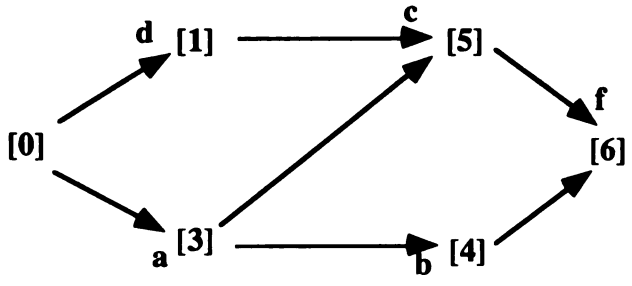
It would be nice if we could use A_M not only as a synthesizer of the structure common to strings in a cluster, but also as a recognizer of other strings which exhibit such a structure. In other words, instead of recognizing whether a string is a common subsequence to the cluster, let us use A_M to recognize if a string has a subsequence which is a maximal one in A_M. We may try the following interpretation procedure: while checking some input, if in state [i] A_M reads a letter α, either there is transition d([i], α) then one proceeds to the following state, or there is no such transition then one reads the following letter. The input is accepted if it is finished being read in a leaf state (one without outgoing arc), otherwise it is rejected. This procedure is sound but not complete: it accepts only strings which have as a subsequence a maximal one in A_M, but not all of those. For example in the above automaton we'll accept rightfully string "eacbdfc" (in state [6] with the maximal subsequence "abf") but we'll reject wrongfully string "edce" although it has the maximal subsequence "de" (in state [5], because of the subsequence "dc"). To overcome this incompleteness we can rely on a non-deterministic interpreter: at a transition d([i], α) the interpreter duplicates itself, one version proceeds normally to the next state and one version stays in state [i] and reads the next letter in the input. Successful termination is when a version accepts the input. Notice that this procedure gives all the maximal subsequences of the input (for "eacbdfc" it provides "abf" but also "acf")

The reader has certainly linked this interpretation procedure of automaton A_M with the chronicle recognition procedure of previous section: in both cases we skip letters (events) in the input stream that are irrelevant for the model at hand, and we manage a tree a partial recognition hypotheses. Can we consider A_M as a skeleton for a chronicle model describing a cluster of observed examples ? Not yet. Although both structures appear to be quite close (partial orders on events), there is a major difference: A_M is a *disjunction* of maximal subsequences whereas a chronicle is a *conjunction* of equivalent subsequences. This is taken care of in the next step.

A maximal subsequence in A_M is characteristic of the strings in a cluster (it appears in all these strings), but it is not necessarily discriminant: it may account for counter-examples as well. Hence A_M is not a robust recognizer. Let

us look for a conjunct of maximal subsequences which rejects all counter-examples. This is done by checking with automaton A_M each counter example in turn and by associating to each maximal subsequence the counter examples it account for. One may construct a lattice of conjuncts (a rooted and/or graph) starting from single subsequences as its leaves. Those which account for all counter-examples are useless and will be removed. Those which reject all counter-examples are kept as discriminant conjuncts. The rest are checked pairwise, removing pairs which do not improve the discrimination and keeping pairs which are discriminant conjuncts. This can be pursued for remaining pairs over triples, etc, until all discriminant conjuncts are found. In principle we do not need all discriminant conjuncts of maximal subsequences, just one is sufficient, since it accounts for all examples in a cluster and rejects all counter examples. However, in practice one may require the selection of few "good" conjuncts which improve the robustness of the recognition. We considered some additional criteria (such as the distance of a conjunct to counter-examples), but we have no results yet to report on that selection issue.

In the example of the A_M above assume that "acf"∧"dcf"∧"abf" is a discriminant conjunct of maximal subsequences. The reduction of A_M to this conjunct is given below. The recognition of a string with this final automata should now proceed exactly as the recognition of a chronicle, since we require all elements of the conjuncts to be subsequences of the accepted string. In other words, we require the 5 letters (or events) a, d, b, c, and f to appear in the observation according to the specified partial order.



We can take this final automata as a skeleton for a chronicle model corresponding to the cluster of examples. We need to enrich it by numerical temporal constraints. These can be obtained by going back to the recorded sequences of events from which the strings of cluster have been abstracted. Remember that these events are time-stamped. For each sequence of events we compute the temporal distance between its significant events, as recognized in the final automata (it has at least one, and eventually several recognized instances). The minimum and maximum distances provide lower and upper bounds of the temporal constraints between two events. A chronicle model with such constraints accepts all recorded sequences of the cluster it represents. But this can be over selective with respect to unmet cases.The final step of the approach relies on the programmer. Interactively, the chronicle model found so far is displayed. The programmer may reject it (in that case another discriminant conjunct of maximal subsequences of the cluster is looked for) or he may accept it. Here the procedure should query the user for an upper and a lower bound of the total duration of the chronicle. Those bounds should be larger or equal to those computed from the examples (otherwise some examples will not be accounted for), they are used to relax uniformly the constraints on the temporal distance between the events in each path. The programmer has also to enrich the model with context information (*hold* predicates)

To summarize the approach, sequences of events are first extracted from recorded data selected and labeled by the user. These sequences are reduced to strings. A sample is clustered into a subset of close strings with respect to the *lcs* measure. To each cluster is associated an automaton AM of maximal common subsequences. A conjunct of maximal subsequences discriminative with respect to counter examples is extracted from A_M. This provides the skeleton of a chronicle model, to which temporal bound from recorded sequences are further added. The result is interactively checked out and augmented by the user.

But of the first and last interactive steps, this approach has been implemented and tested [Marois 95]. A file of simple synthetic examples containing a hundred sequences of about 30 events each from an alphabet of 20 event types is processed in less than a minute. The costly step of extracting automaton AM from a cluster of 30 strings (about 60 states and 100 maximal subsequences) takes in the order of a second. These figures (not intended as significant measurements) are indicative of the feasibility of the approach given the fairly short strings we'll be concerned with.

5. CONCLUSION

This paper proposed a knowledge representation which enables to model in a fairly generic way temporal scenarios whose occurrences need to be recognized by a supervision system. This representation admits some restrictive assumptions, e.g., solely conjunctions of events, assertions and constraints within a chronicle. These assumptions were not found too restrictive in practice. In fact, the main restriction is that the completeness of the chronicle recognition requires the completeness of the event detection. Notice that the plan recognition work or the abductive diagnosis approaches referred to earlier [Console 92, Nejdl 94, Rao 94] make the same assumption. But one has seldom a such reliable

observation system, although at the level of symbolic events we are concerned with here, comfortable robustness is achievable. We need to extend the representation and the robustness of the approach by taking into account general domains knowledge expressing dependency between fluents to enable cross checking and eventually the inference of accidentally non observed events.

We described here an efficient real-time recognition system for the proposed chronicle representation. Giving our assumptions, this system is able to manage a complete hypothetical reasoning for all instances possibly taking place. This system reads its input flow only once; it does not manage a working memory in the classical sense since predicts expected event instances and their windows of relevance, and hence it never has to backtrack. The system has been field-tested in real industrial applications. It exhibits good performances.

Finally we proposed a learning approach for extracting chronicle models from examples. This approach considers chronicles as a particular case of automata. It relies on the large body of algorithms for synthesizing finite automata. We have adapted and extended few of them for our purpose. Although this learning method has not yet been field tested as the recognition system has, it seems promising and it enhances the interest of the chronicle representation.

Acknowledgments: this work has been conducted with my students A.Mounir-Alaoui, C.Dousson, P.Gaborit and F.Marois.

REFERENCE

Bernard D., Borillo M. and Gaume B. From event calculus to the schduling problem. *Journal of Applied Intelligence*, 1:195-221, 1991

Console L., Portinale L., Theseider D. and Torasson P. Diagnostic reasoning across different time points. Proc. 10th ECAI, p.369-373, 1992

Dechter R., Meiri I. and Pearl J. Temporal Constraint networks. *Artificial Intelligence*, 49, p. 61-95, 1991

Dousson C. et Ghallab M., Suivi et reconnaissance de chroniques. *Revue d'Intelligence Artificielle*, 7 (2), 1994

Dousson C., Gaborit P., Ghallab M., Situation recognition: representation and algorithms. Proc. 13th IJCAI, 1993

Ghallab M. and Mounir-Alaoui A. Managing Efficiently Temporal Relations through Indexed Spanning Trees, Proc. 11th IJCAI, 1297-1303, Detroit, 1989

Guenoche A. and Vitte P. Méthode exacte et approchée pour le problème de la plus longue sous-séquence commune à p chaines de charactères. Tech.Report, Univ. Aix-Marseille2, 1992

Hebrard J.J. Distances des mots, Applications à la recherche des motifs. Thèse de Doctorat, Univ. de Rennes, Février 1984

Hunt J.W. and Zymanski T.G. A fast algorithm for computing longest common subsequences, *Comm. ACM*, 20(5):350-353, 1977

Kowalski R. Database update in the event calculus. *Journal of Logic Prgramming*, 12:121-146, 1992

Kumar K. and Mukerjee A. Temporal event conceptualization. Proc. 9th IJCAI, p.472-475, 1987

Mackworth A.K. and Freuder E.C. The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artificial Intelligence*, 25, p. 65-74, 1985

Marois F. Apprentissage de scénarios. LAAS report 95390 (Masters thesis) June 1995

Milne R., Nicol C., Ghallab M. et al. TIGER: a real-time situation assessment of dynamic systems. *Intelligent Systems Engineering*, 3:103-124, 1994

Mounir-Alaoui A., Raisonnement temporel pour la planification et la reconnaissance de situations, Thèse de l'UPS, Toulouse, 1990

Nebel B. and Burckert H.J. Reasoning about temporal relations: a maximal tractable subclass of Allen's interval algebra. *J. ACM*, 42(1):43-66, 1995

Nejdl W. and Gamper J. Harnassing the power of temporal ctoins in modelbased diagnosis of dyna:ic systems. Proc. 11th ECAI, p.667-671, 1994

Oncina J. and Garcia P. Inferring regular languages in polynomial updated time. In Proc. 4th Spanish Symposium in Pattern Recognition and Image Analysis, 1992

Rao A.S. Means-end plan recognition - Towards a theory of reactive recognition. Proc. 4th KR, p. 497-508, 1994

Shoham Y. *Reasoning About Change: Time and Causation from the Standpoint of Artificial Intelligence*, MIT Press, 1987.

Vilain M. and H. Kautz, Constraint Propagation Algorithms for Temporal Reasoning, Proc. AAAI, 1986.

Psychological and Philosophical Connections

Psychological Constraints on Plausible Default Inheritance Reasoning

Carl Vogel & Judith Tonhauser
Institute for Computational Linguistics
University of Stuttgart
Azenbergstr 12
D-70174 Stuttgart, Germany
{vogel,tonhauser}@ims.uni-stuttgart.de

Abstract

Human responses to a large number of problems that are relevant to the literature on default reasoning are examined. Two foundational assumptions of the literature are identified: chaining of defaults is possible; chaining is not possible beyond a negative default. These assumptions are used as *strict* criteria to select subjects who always behaved in accord with them. Their within-subject patterns of response are examined. The paper identifies 5 reasoning strategies: skepticism, explicit link acceptance, shortest path reasoning, most paths reasoning, and a combination of the latter two reasoning. Of these, the literature has hitherto regarded seriously only extreme skepticism; for example, shortest-path reasoning has been discounted since (Touretzky, 1986). The strict inclusion criteria are then relaxed, and over a space of consistent reasoners the same reasoning strategies remain useful in classifiying human response. If we want machines to reason about defaults the way people do, or reason about people who reason with defaults, the field must address these easily formalized modes of reasoning.

1 INTRODUCTION

Hierarchies are ubiquitous in native human organization of information and often contain tangles and defaults (e.g. conceptualizations of social/honorific hierarchies). Path-based default inheritance reasoning is interesting as a form of nonmonotonic inference with tangled hierarchies because of its attractive computational properties. Inheritance can be viewed as a variant of the monadic predicate calculus, with a guarantee of inferential acyclicity, and for this reason should always be decidable, sometimes tractable.[1] For example, the skeptical theory of inference proposed by Horty et al. (1990) is polynomial, and of course more restricted systems perform even better (Niemelä & Rintanen, 1994). However, efficiency is not always a property of inheritance reasoners (Selman & Levesque, 1989, 1993) and certainly not of nonmonotonic inference at large. Complexity is introduced in the varying ways that inference can be defined on the acyclic graphs isomorphic to inheritance theories. There are, of course, any number of ways of defining this inference, and competing methods appeal to 'intuitions' about which methods are most appropriate (Touretzky, Horty, & Thomason, 1987). The intuitions at stake are those which define the most plausible conclusions on the basis of particular interpretations of graphs for those graphs which different conclusions depending upon the reasoner applied. Arguments about plausibility, rather than validity, are made because inheritance is designed to be a psychologically plausible model of human reasoning with defaults.

Inheritance reasoners purport to provide a psychologically plausible model of reasoning with defaults, partially motivated by the ubiquity of tangled hierarchies themselves in the organization of information, and partially because the efficient decision procedures associated with the representation make it seem a reasonable descriptor of human reasoning, which is relatively efficient. Until very recently there have been no psychological investigations designed to elucidate the semantics of reasoning with generics with respect to the idealizations of inheritance theory. Elio and Pelletier (1993) present results about the way people classify exceptional objects in light of default theories in relation to the way general default logics classify the same exceptional objects. They also present the first pilot study applying similar scrutiny to inheritance reasoners. Hewson and Vogel (1994) and Vogel (1995) directly test the plausibility of inheritance reasoning, in an attempt to assess the degree of fit that popular inheritance reasoners (like that of Horty et al. (1990); hereafter called H90) have with the data supplied by human reasoning. This paper describes an extension of that work.

[1] Thanks to Jeff Pelletier for making that clear to us.

The data from their experiments is pooled into a substantial body of evidence on human reasoning with sets of defaults. Rather than assessing the degree of fit of extant systems with the data (since none of the path based systems fit very well), observations are made concerning the features of reasoners which would provide a good predictor of human reasoning, and a set of reasoners derived from these observations are characterized. We proceed by outlining our assumptions and characterizing the experiments from which the data analyzed here are drawn. Then we define a consistency criterion with respect to satisfying transitivity and negative reasoning which we apply strictly to select those subjects who were most consistent in their overall replies, and outline the reasoning strategies which are predictive of their responses. We then provide a similar analysis for subjects grouped according to a range of degrees of satisfying the inclusion criterion. The same basic patterns are apparent at each stage: primarily full skepticism, and among the nonskeptics, a mixture of a kind of shortest-path/most-paths reasoning.

1.1 DEFINING PLAUSIBILITY

The psychological plausibility of a logic can be defined as the degree to which it captures the reasoning patterns that people ordinarily use. One can define logics of 'ideally rational agents'; however, those logics are nearly always undecidable, and therefore are far from ideal for a rational agent to be driven by them. Goldman (1986) also argues that the concept of rationality should be dictated more by what human behavior demonstrates, instead of rating human reasoning as defective. We are not immediately concerned with whether logics provide a correct morphism to the processes that actually govern human reasoning; rather, our interest is the sort of logic that provides the best description language for expressing exactly those inferred sentences that people are likely to agree are true or worth acting upon. If one's goal is to describe human reasoning, then this is a reasonable way of proceeding. If the goal is to develop a machine that reasons more correctly than humans, *if it is to interact with humans it will require a model of human reasoning.* From this perspective, plausibility can be measured in terms of the degree of fit between the conclusions licensed by a logic about a set of premises, and the conclusions reached by people. Depending upon the logic, there may be correspondences between proof theoretic parameters and reasoning strategies that people use (consistently or in different circumstances).

1.2 TESTING PLAUSIBILITY

Hewson and Vogel (1994) addressed inheritance reasoning directly by attempting to determine what sorts of conclusions people find in sets of abstract defaults. The statements in the sets were abstract in the sense that uninterpreted roman letters were used as labels for concepts (*Bs are normally Cs*) rather than given concrete interpretations (*Marines are normally short haired*), to allow control over the influence of other knowledge, belief or opinion (Kaufmann & Goldstein, 1967). Subjects were supplied five possible answers to each problem, with appropriate instantiations for X and Y depending on the exact problem: *Xs are normally Ys*; *Xs are normally not Ys*; *Xs are normally Ys and Xs are normally not Ys*, *it is indeterminate whether Xs are normally Ys or normally not Ys*, *I don't know.* The materials were questionnaires comprised of 40 problems, each problem containing a set of statements constituting a default theory about abstract concepts. Subjects were asked to give the answer they thought appropriate on the basis of the information given in each problem alone—it was stressed that there were no right or wrong answers, and subjects were to say what *they* thought could be concluded. Materials were supplied with the default theories in three modes of presentation: sentence only, graph only, sentence and graph. Each of 72 subjects (in a range of ages and generally with non-technical backgrounds) received all 40 questions (in a random order or the reverse) in one mode of presentation. The 40 problems were designed to encompass a number of inheritance networks specifically disputed in the literature in relation to the conclusions they should 'intuitively' license, the choice of answers constrained by the sorts of classifications of the problems made in the the literature. An example problem from the graph+sentence mode of presentation is given in Figure 1. The problems were comprised solely of defaults. In the graph-only condition, subjects were told only in the instructions that the arrow meant 'are normally', and thus had substantially less priming of 'defaultness' than subjects in sentential conditions. It is convenient to index each problem by its graph presentation. In each problem, the respondent is asked to characterize the relationship between the 'leftmost' and 'rightmost' class mentioned in the premises.

Some problems were set to test the more foundational assumptions of the inheritance literature: transitivity, negativity, redundancy and preemption. The remaining problems were controls. Hewson and Vogel (1994) reported surprising findings. Firstly, most of the problems were rated indeterminate.[2] Of the determinate response, while there was considerable support for transitivity, and some support for simple cases of preemption, there was little evidence for redundancy (redundant statements/links seemed to be interpreted as providing additional information rather than re-

[2] Answers in categories 'c' or 'd' are referred to as *indeterminate classification* of a problem. An answer in category 'a' is *positive classification*; 'b', *negative classification.* An answer of either 'a' or 'b' is a *definite classification* or *determinate.*

1

- As are normally Bs.
- Bs are normally Cs.

What can you conclude from these statements? Asterisk (*) the appropriate answer.

(a) As are normally Cs.
(b) As are normally not Cs.
(c) As are normally Cs and normally not Cs.
(d) It isn't definite whether As are normally Cs or normally not Cs.
(e) I don't know.

If you wish, explain why you reach this conclusion.

Figure 1: An Example Question

dundant and disregardable information), and people seemed to reason with negative information in a way wholly unpredicted by the inheritance literature (Vogel, 1996). Hewson and Vogel (1994) did not report evidence about the various specific forms of preemption or subpath ambiguity. There was no clear finding with respect to mode of presentation, except that graphic presentation seemed to polarize responses more, suggesting that the graphic syntax of inheritance reasoning has much to do with the strength of 'intuitions' that have been discussed in the literature. Vogel (1995) partially replicated the experiment of (Hewson & Vogel, 1994), dropping the graph+sentence condition and testing the materials on an additional 98 subjects (undergraduates in English literature and composition courses); the results corroborated the earlier findings.

This paper pools the data from the 2 experiments. After removal of unattempted questionnaires, and those for which collation errors during material preparation resulted in different questionnaires than other subjects had (due to repetition & omission of sheets), there are 162 questionnaires in total. With 40 questions each there were potentially 6480 datapoints. However, altogether there were 71 unanswered problems, leaving 6409 datapoints in total. This is a substantial body of data which is open to a great deal more analysis than Hewson and Vogel (1994) or Vogel (1995) supplied. Here we will examine particular descriptors of response patterns that can be articulated as reasoning strategies. For discursive convenience, this paper will refer to the *descriptors* of reasoning patterns as if they were *strategies* that determined responses. Thus, this paper outlines three nonskeptical inference strategies that were consistently 'employed'. The strategies are identified in examining patterns of responses within subjects to sets of problems from the described experiments and sets the groundwork for examining their between subjects systematicity. These strategies are then taken as formal definitions of inheritance reasoners and the predictive efficacy of each will be evaluated.[3]

2 PSYCHOLOGICALLY PLAUSIBLE REASONERS

In this section we examine response patterns of subjects who satisfied criteria of regularity in responses to related problems. We initially adopt absolute consistency criteria relative to basic assumptions of inheritance reasoners and then examine the results among subjects who satisfied these criteria in gradual degrees. The hope is that by identifying such systematicity in response patterns we can identify or define reasoners which yield the closest approximation to observations. We consider within-subject patterns of reasoning (the previously mentioned analyses considered only between-subject systematicity for individual problems; the current work attempts to assess between subject systematicity over *sets* of related problems). There are two rational patterns of reply which we considered. The first was dictated by basic assumptions of the literature and the second by systematic behaviors which actually appeared. For both patterns, under the absolute criteria, data obtained from subjects who did not conform to predictions of positive and negative transitivity (measured by problems containing no conflicts) were eliminated; this left 88 questionnaires. This is motivated by the consideration that the reasoning of those subjects whose responses were not predicted by transitivity on graphs containing no conflicts would be better predicted by classical statistical models (and are thus outside the scope of default inheritance) or were otherwise relatively inconsistent in response patterns. The latter possibility is examined in § 2.2.

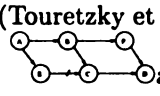
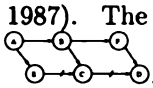
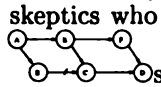
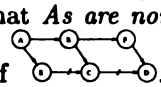
2.1 ABSOLUTE CONSISTENCY

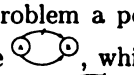
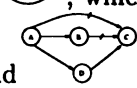
2.1.1 Negative Chains are Indeterminate

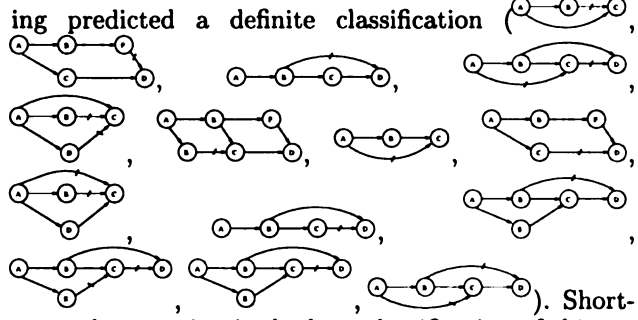
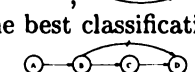
In the first absolute inclusion criteria, additional subjects whose responses to problems containing non-final negative links (e.g. ⊙⇢⊙—⊙) propagated negativity (i.e., concluded definitely that *As* are normally not *Cs*)

[3] An informal aside on complexity: interesting (but inconclusive) facts about the time it took subjects to complete the experiments (we emphasize that these were not reaction-time experiments), conjoined with estimates on average reading time (e.g. Rayner & Pollatsek, 1989), suggest that linear reasoning methods (e.g. Niemelä & Rintanen, 1994) are too efficient and that polynomial methods (e.g. Horty et al., 1990) are just about right.

were also eliminated,[4] leaving the materials of 8 subjects. While this is a rather small resulting dataset, it should be pointed out eliminations were taken to be stringent: in the transitivity case, subjects were eliminated if they did not respond in accord with transitivity in each of the 8 graphs without conflicts; in the negativity case, subjects were eliminated if they propagated negativity with any of the 7 (different) graphs involving nonfinal negative links.[5] In this initial analysis we were less strict with respect to negativity cancellation than to propagation, in homage to the fact that humans find negative reasoning in general more difficult: if a subject gave a positive classification to at most one problem involving double negatives, we still admit them into this analysis. This resulted in the inclusion of one subject; thus, the inclusion criteria remain quite strict. A larger number of subjects satisfied the inclusion criteria to lesser degrees. In the context of the detailed within-subjects analysis, extreme stringency is most appropriate since it still illustrates the main points. Clear reasoning strategies emerge and are satisfied by other subjects as well. The eliminations made are both justified in that they leave only those subjects who are at least in accord with the foundational assumptions of default inheritance reasoning. The subsequent section (§ 2.2) examines subjects satisfying the inclusion criteria to varying degrees. The same basic reasoning patterns emerge among the additional 43 subjects as we find here among the 8.

Of the 8 subjects who satisfied the strict inclusion criteria, 3 were more or less credulous.[6] The remainder, with 2 exceptions, rated graphs as indeterminate unless there were no conflicting paths. In terms of the literature, they were *skeptical* in the face of conflicting defaults, yet still accepted chaining of uncontested defaults (which is classically invalid). The exceptions in this remainder are rather interesting, because they involved positive classification of 2 problems which are canonical for the dispute over whether ambiguity in a subpath should be cascaded (Touretzky et al., 1987). The graphs in question are  and  . To the former, which the literature assumes to contain no conflicting paths because of the status of propagation of negativity, three of the 'skeptics' responded by giving it a positive classification: *As are normally Ds*. The other 2 skeptics classified the graph as indeterminate. The latter graph was classified positively by one of the skeptics, and as indeterminate by the other four. Curiously, none of the skeptics who determined that *As are normally Ds* of  said the same of  . This means that the five skeptics were partitioned into: the one who classified both as indeterminate, the three who found the first one positive, and the one who found the second one positive. As far as adjudicating inheritance proof theory is concerned, this would argue for both forms of skepticism, with the restricted skepticism of Horty et al. (1990) fitting 25% of the time. Of the three 'credulous' reasoners, 2 classified both of the problems positively. Given that 'credulous' is being used in a rather nontechnical sense here (referring not to the sanctioning of a multiplicity of extensions, but to likelihood to make some consistent and definite classifications of problems) this actually means that there was equal support for both ambiguity propagating skepticism and the non-propagating restricted skepticism of Horty et al. (1990). No wonder there is a clash of intuitions.

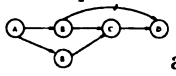
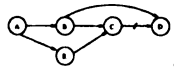
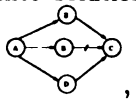
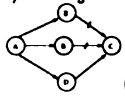
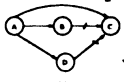
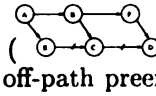
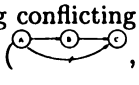
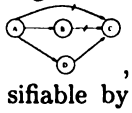
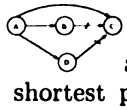
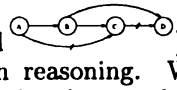
The 3 subjects who made more definite classifications than the skeptics supply interestingly consistent responses. Subject #62 was the most credulous. With 2 exceptions, the response strategy of #62 would most aptly be labeled *affirmation*: if a positive path existed #62 gave the problem a positive classification. The 2 exceptions are  , which #62 classified as indeterminate ('c') and  (the nixon diamond with a direct negative link), which #62 classified as negative. However, the remaining 32 problems which did not involve a nonfinal negative link in a single chain of links were all given definite classifications, and the only other negative classifications among these 32 were of those problems in which there was no positive paths available.

Subject #12 was the next most credulous of the 3 in the sample under focus. Subject #12 favored the response of shortest path reasoning where conflicting paths existed and shortest path reasoning predicted a definite classification (). Shortest path reasoning is the best classification of this response strategy because,  was evaluated

[4] We refer to chains containing nonfinal negative links as *negative chains*.

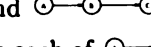
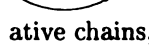
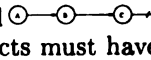
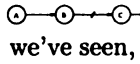
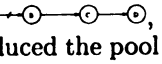
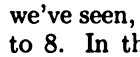
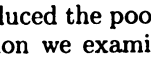
[5] If a chain contained two negative links, it would be possible for the negativity to either intensify, yielding a definite negative classification, or to cancel, yielding a definite positive classification.

[6] 'Credulous' in the sense of the inheritance literature, means willingness to draw definite conclusions where others might be skeptical and draw none (e.g. faced with ambiguity).

by this individual as indeterminate, and this particular problem is the canonical example of why simple shortest path reasoning is not desirable (Touretzky, 1986), leading to the definition of more complex path preference strategies (which classify this problem negatively using answer 'b'). Subject #12 agreed with the shortest path prediction for this problem by classifying it as indeterminate. Interestingly, this pattern of classification also holds for 2 problems canonical for the dispute between on-path and off-path preemption:  and  . Subject #12 classified both as a shortest path reasoner would. As it happens, this is the same classification predicted by off-path preempting reasoners like Horty et al. (1990) and Al-Asady and Narayanan (1993). This is interesting because off-path preempted chains can be seen as a species of 'redundant' links, and in the simple case this subject did not disregard the redundant information. However, in the simple case the 'redundant' link rendered the problem indeterminate with respect to shortest path reasoning. In response to problems which shortest path reasoning did not resolve to a definite solution, Subject #12 gave a *most paths* reply:  ,  (note that  was classified with answer 'a'). Thus, Subject #12 seems to be representative of the following reasoning strategy: shortest path where that makes definite conclusions available, most paths where that makes definite conclusions available if shortest path reasoning doesn't, and indeterminate otherwise. Given this classification of the responses of Subject #12, it is interesting in addition that this subject classified () positive determinate, consistently with off-path preemption and non-cascaded ambiguities. Moreover, this subject was in a sentence only condition, thus exhibiting remarkably complex consistency for problems whose presentation did not directly support graphical classification; thus, this subject stands as an argument for the predictive efficacy of some aspects of path-based inheritance reasoning. Subject #51 was the most skeptical of the credulous respondents. The four problems involving conflicting arguments which #51 classified definitely ( ,  ,  and ) were all classifiable by shortest path reasoning. While shortest path reasoning seems to be the simplest description of the strategy at work, it is interesting to note that in 3 of these cases, the shortest path was only one link long.

It is worth pointing out that the *affirmer* participated in the graph-only condition of the experiment, the *shortest path reasoner* participated in the sentence-only condition, and the *explicit link acceptor* (#51) was in the sentence+graph condition. Of the 5 skeptics, one was in the sentence-only condition; another, sentence+graph; the other 3, graph-only. This is an interesting distribution with respect to the distribution discussed in the next section and the overall inconclusive mode-of-presentation findings of Hewson and Vogel (1994). Only in the most consistent category are there so few subjects who were in the sentence-only condition. This is the clearest evidence to date that the conditions that offered graphical stimuli polarized response patterns. This is intuitively to be expected, since the sentence-only condition is quite difficult without visualizing the problems.

2.1.2 Negative Chains are Negative

In the last section we reported the within and between subjects response patterns among the 8 subjects whose behavior was most consistent, and which respected certain foundational assumptions of inheritance reasoning: we ruled out subjects who did not answer according to the predictions of transitivity in problems without conflicts, as well as those subjects who did not classify negative chains as indeterminate. For the first criterion subjects must have 'correctly' classified *each* of the 'valid' transitivity problems with *determinate* classifications, making positive definite classification of :  ,  ,  , and  , as well as answer category 'b' to each of,  ,  ,  , and  . In the case of negative chains, subjects must have classified *each* of the following problems as *indeterminate*:  ,  ,  ,  ,  ,  , and  . As we've seen, this reduced the pool of 162 subjects down to 8. In this section we examine the data admitted by an absolute relaxation of the negativity criterion. This is the approach taken by Vogel (1996) who found most paths reasoning to be a robust predictor.

First we describe the relaxation of the negativity criterion. It is a basic assumption within the inheritance literature that a negative link can occur only in a path-final position. A chain of links with a nonfinal negative link is a negative chain. In general, there is no valid information to be had about a definite relationship existing between the endpoints of such a chain. Thus, rating negative chains as conveying definite negative information is a different sort of divergence than labeling some cases of uncontested transitivities as indeterminate. However, it should be no surprise that negative reasoning is problematic, this result often appears in psychological investigation of human reasoning, for instance Wason and Johnson-Laird's (1972) card selec-

Problem	Response a	b	c/d
3	3	13	15
5	12	2	18
6	5	12	15
7	15	3	14
8	3	14	14
17	15	1	16
22	1	10	20
26	1	13	18
31	17	0	15
33	1	11	19
35	1	12	16

Table 1: Between-Subject Patterns of Response to Problems

tion task, and Evans' (1983) model construction task. The fact is that people are not very good at negative reasoning. Our conclusion, given that we think we should be interested in modeling human rationality, is that invalid negative reasoning involves rational, if pragmatically suspect, strategies. Given that we want to develop models of human reasoning, we should have a model which is inclusive of objective flaws in human reasoning.

Vogel (1996) analyzed the data using exactly this relaxation (*all* of the transitivity cases as predicted, and *all* of the negativity cases contrary to the literature's prediction) and thus obtained a pool of 32 subjects.[7] Table 1 depicts a between-subjects summary of responses to 11 problems for which significance obtained. Results for an additional 13 problems for which significance did not obtain are excluded here, one other problem which could have been used as an inclusion criterion was also excluded. Refer to Vogel (1996) for the full table. Table 1 shows that there is mainly indeterminacy, and in many cases a roughly equal amount of determinacy. For discursive convenience, call the problems with a preponderance of 'a' responses (5, 7, 17, and 31) among the determinate replies the *A-Problems*. The problems with significant numbers of 'b' classifications are 3, 6, 8, 22, 26, 33 and 35. Call those the *B-Problems*. Note that when negative chains are allowed to be classified as providing definite negative reasoning, then the pattern of reply on these problems fits with the predictions of most path reasoning.

A within-subject study of the data revealed that 11 subjects classified all of the A-Problems positively and all of the B-Problems negatively. That is, a full third of the subset were rather consistent internally. Another 3 subject did not classify all 4 of the A-problems positively, but did classify all of the B-problems negatively, and a further 6 subjects classified most of the A-problems affirmatively and tended to classify the B-problems negatively as well (3 or more). Thus, two-thirds of the subjects in this subset were in fact largely consistent with respect to the predictions of most paths reasoning in these cases. It is important to emphasize that the subjects were drawn from both the graphical conditions and the sentence-only conditions with the same patterns of reply in both of those conditions. Thus, it would appear that at least within this partition of subjects, with the allowance of negative chains as negative paths, most paths reasoning is the best predictor of replies. Note that the replies to problem 5 is counter to what is ordinarily predicted by the literature on path-based inheritance, as well as most paths reasoning.

2.2 DEGREES OF CONSISTENCY

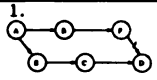
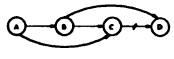
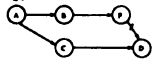
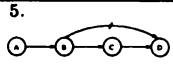
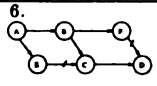
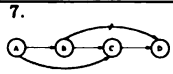
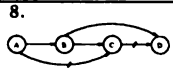
We have reported the within and between subjects response patterns among the subjects whose behavior was most consistent with respect to transitivity. We also considered two different absolute inclusion criteria regarding negative reasoning: the first (and most prolifically exclusive) was the assumption that negative chains are indeterminate; the second, that negative chains are negative. With a consistent pattern of reply, both patterns are rational. In the first case, the subject pool was reduced from 162 to 8, and in the second to 32. We reported the patterns of definite response to problems not among the inclusion criteria and identified potential reasoning strategies which would predict those patterns.

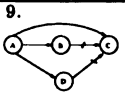
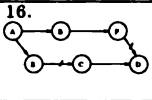
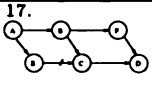
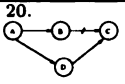
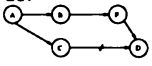
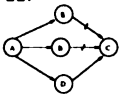
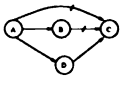
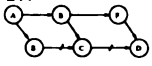
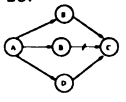
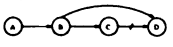
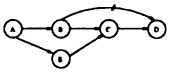
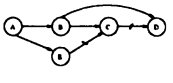
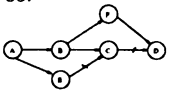
However, the patterns of response are in general systematic, even among subjects who were less consistent over the whole questionnaire. In this section we examine the data admitted by relaxing the inclusion criteria and demonstrating that the same patterns of response hold in the larger pool of subjects, albeit with increasing inconsistency of within-subject predictability corresponding to increasing inconsistency with respect to the inclusion criteria. There are three basic systematic strategies that can be used to do this. One approach is to hold fixed the idea that negative chains are indefinite and consider degrees of satisfaction of transitivity. Another is to hold transitivity fixed and

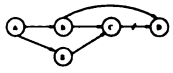
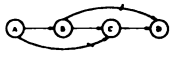
[7] By the definition of the inclusion criterion, these are exclusive of the 8 subjects discussed above.

consider increasing degrees of definite response to the negative chains (the extreme point of the latter relaxation yields exactly the subjects considered in the last section, and the minimally inconsistent along the former relaxation are exactly those subjects considered in §2.1.1). The third approach is to consider degrees of satisfaction on both axes of inclusion simultaneously.

At this point, since more problems will have definite answers as we progressively relax the inclusion criteria, it becomes useful to tabulate the 25 noninclusion problems along with the categories of response appropriate to the main reasoners we have addressed here: explicit link acceptance, shortest path reasoning, most paths reasoning. For comparison, we also give the answer to each problem according to H90 and to a reasoner just like H90 except more skeptical, using on-path preemption and not discounting paths with nonpermitted subpaths. The headings on the following list — **EL**, **SP**, **MP**, **H90** and **SK** — correspond to those reasoners. As we have also considered the possibility of negative chains being either indeterminate or negative, we mark responses made by a reasoner with an annotation indicating if its answer depends on negative chains being negative ($*^-$) or on them being indeterminate ($*_?$). The following table summarizes this information. Note that in some cases the reasoners are in agreement. In the analysis which follows we make within subject examinations of consistency in determining which strategies are most often representative.

Prob.	Resp.	EL	SP	MP	SK	H90
1. 	a					
	b					
	c/d	*	*	*	*	*
2. 	a	*	*		*	*
	b					
	c/d			*		
3. 	a				*	*
	b			*		
	c/d	*	*			
4. 	a		*			
	b					
	c/d	*		*	*	*
5. 	a					
	b		*		*	*
	c/d	*		*		
6. 	a					
	b			$*^-$		*
	c/d	*	*	$*_?$	*	
7. 	a			*		
	b				*	*
	c/d	*	*			
8. 	a		$*_?$		*	*
	b			$*^-$		
	c/d	*	$*^-$	$*_?$		

Prob.	Resp.	EL	SP	MP	SK	H90
9. 	a	*	*		*	*
	b			*		
	c/d					
16. 	a					
	b		*	*	*	*
	c/d	*				
17. 	a		$*_?$	*	*	*
	b					
	c/d	*	$*^-$			
18. 	a					
	b	*	*		*	*
	c/d			*		
20. 	a					
	b					
	c/d	*	*	*	*	*
21. 	a					
	b		*			
	c/d	*		*	*	*
22. 	a					
	b			*		
	c/d	*	*		*	*
26. 	a					
	b	*	*	*	*	*
	c/d					
27. 	a					*
	b			$*^-$		
	c/d	*	*	$*_?$	*	
28. 	a			*		
	b					
	c/d	*	*		*	*
29. 	a		*		*	*
	b					
	c/d	*		*		
31. 	a			*		
	b		*			*
	c/d	*			*	
33. 	a		*		*	*
	b			$*^-$		
	c/d	*		$*_?$		
35. 	a					*
	b			$*^-$		
	c/d	*	*	$*_?$	*	

Prob.	Resp.	EL	SP	MP	SK	H90
38. 	a					
	b					
	c/d	*	*	*	*	*
39. 	a		*			*
	b			*		
	c/d	*			*	
40. 	a					
	b		*	*⁻	*	*
	c/d	*		*?		

2.2.1 Relaxing Transitivity

For the present analysis, we considered subjects whose response patterns were less consistent with respect to the transitivity requirements. We examined the patterns of response over the remaining problems which were given definite classifications. That is, we held fixed the negativity requirement from the stringent consistency criteria in which negative chains are indeterminate, but analyzed data from subjects who did not answer as predicted on all of the transitivity cases. This admitted another 8 subjects; however, inspection revealed that 3 of them did not get any of the transitivity cases as predicted, and 1 got only 2. Their materials were hard to find patterns in. Of the remainder, 3 subjects answered 6 of the transitivity problems according to the criteria and 1 of them answered 3 as predicted. It is those 4 subjects (#18, #54, #57, #81) whose patterns of reply we examine next.

Subject #81 was the most skeptical of these 4. This subject gave only 5 definite classifications among the 25 problems outwith the inclusion criteria. Of these, one classification was in accord with the only path in the network under the assumption that the negative chain was deemed indeterminate (an assumption that is, of course, consistent with the inclusion criteria assumed in this section). Another 2 problems were consistent with explicit link acceptance, and 2 more with shortest path reasoning. Thus, the behavior of this individual in problems classified definitely is best predicted by shortest path reasoning. However, we lack a classification of problems for which shortest path reasoning or explicit link acceptance could have been used, but wasn't, by this individual.

Subject #57 gave 13 definite responses; 4 of them were in accord with an explicit link in the problem, and the remaining 9 classified by shortest path reasoning. Similarly, subject #18 gave 12 definite replies, 4 of them agreeing with explicit link acceptance, another 3 consistent with shortest path reasoning. Two of this subject's responses were in accord with the polarity of the only path through the network under the inclusion assumption regarding negativity, and 3 additional definite replies could not be classified by any reasoning strategy we have so far considered. Finally, subject #54 gave 3 answers agreeing with explicit link acceptance, 4 agreeing with shortest path reasoning, 4 agreeing with most paths reasoning, 3 which were in accord with the polarity of the only path given the inclusion assumptions, and 1 more which was not predicted by any strategy we've addressed.

Problem	Response a	b	c/d
3	0	1	3
5	0	3	1
6	1	1	2
7	0	0	4
8	3	0	1
17	3	0	1
22	0	1	3
26	0	4	0
31	0	3	1
33	1	1	2
35	0	0	0

Table 2: Transitivity Relaxed, Between-Subject Patterns of Response

Overall among the subjects who consistently acted in agreement with the literature's assumption that negative chains are indeterminate shortest path reasoning (with explicit link acceptance as a special case) seemed to be the most frequent predictor of subject response. Assuming that the classification of negative chains also applied in larger graphs, these subjects tended to agree with transitivity applied to the remaining links. Most path reasoning was a reasonable predictor where shortest path reasoning was not. While this demonstrates that the reasoning patterns identified under the stringent reasoning criteria are also apparent elsewhere in the data where the transitivity requirement is relaxed, this relaxation still generates only a very small pool of subjects to consider.

2.2.2 Relaxing Negativity

Holding transitivity fixed but allowing increasing amounts of determinate responses to negative chains yields 20 subjects. Table 3 encapsulates the between subject patterns of reply for those cases where significance obtains.

A within-subject analysis reveals that there are 9 skeptics. Of these, 5 were already discussed in § 2.1.1, before the relaxation of the inclusion criteria, as well as the affirmer, and the explicit link acceptor. There were 3 whose behavior was consistent with preferring shortest path reasoning in cases in which it could resolve the problem, and most paths reasoning otherwise (one of these was #12, also discussed above). An additional 3 were best predicted by the reverse preference: most paths reasoning and then shortest paths. A final 3 behaved as if guided by most paths reasoning (and are not predicted by the acceptance of explicit links). The

Problem	Response a	b	c/d
2	6	1	14
7	6	0	15
8	7	3	11
9	6	1	14
16	0	17	4
17	11	0	10
18	0	7	14
22	1	5	15
26	1	8	12
28	4	1	16
31	6	2	13
33	5	0	16
35	4	0	17
38	0	0	21
40	1	5	15

Table 3: Negativity Relaxed, Between-Subj. Patterns

average number of definite answers for these subjects is 6.86 (σ = 6.07). Thus, apart from affirmation, we see in this relaxation of the inclusion criteria the same patterns of response that were found among the smaller set of absolutely consistent subjects. Increasing use of most paths reasoning accompanying increased likelihood to classify negative chains negatively.

2.2.3 Degrees of Transitivity and Negativity

For the final set of analyses, we considered simultaneous relaxation of both the transitivity and the negativity requirements. We examine the results according to participation of subjects in groups defined by a certain degree of consistency. We with the exception of affirmation, we find the same representation of reasoning strategies in each group, although with less consistent internal applicability as the groups tend towards greater deviation from the inclusion criteria. The groups, as a function of degree of meeting the inclusion criteria, broke down into 5 classes, encompassing 51 subjects eligible for consideration through being systematic in their responses. The inclusion function was defined from a matrix with increasing intransitivity as one axis and decreasing negative chain indeterminacy as the other axis. The group that a subject falls into is then determined from this matrix.[8] Below we consider variances among these groups.

For these purposes, it made sense to make the strict criterion even more strict, so that the one subject considered above (#12) who gave a positive definite classification to a negative chain with 2 negative links was instead placed in the second group.[9] This means that Group 1 is comprised of 4 complete skeptics, one explicit link acceptor and the one affirmer. Also, this entails that the revised Group 1 is the only group comprised solely of subjects in just the graphical conditions. In the remaining groups there was roughly an even distribution between subjects in the sentence only condition and subjects in the graphical conditions. An exception is Group 2 which had 5 subjects in the sentence-only condition. That so many subjects from this (the most abstract and taxing) condition were in the group second-most consistent with respect to foundational assumptions of inheritance is a concrete support for the rationality of those assumptions. This section proceeds by summarizing the results of an analysis like the detailed one given in above for the Group 1 subjects—we examined the within-subject patterns of response, but report just the between-subject averages of those within-subject patterns.

Group 2. The 6 subjects in this group tended to respond definitely to more problems than subjects in Group 1 (there the average was 5.9 (σ=8.2), apart from the *affirmer* who answered 23 definitely, rather exceptionally for the study as a whole); in Group 2, the average was 10.6 (σ=4.8) definite classifications per subject, beyond those required by the inclusion crite-

[8] If i is the group number, then subject n is in group i if that subject's number of 'incorrect' responses to transitivity and negative chain test cases falls into the region of the matrix defined by the $g(i)$ cells of the matrix closest to the origin uniquely corresponding group i. This function is defined as follows: $g(0) = f(0) = 0$, $f(i) = i^2 + i$, $g(i) = f(i) - f(i-1)$; essentially, this describes a regular series of encompassing rectangles. There are 2 cells in the first group, one empty as there were no subjects with all 7 negative chain problems 'correct' but one of the transitivity problems 'wrong'.

[9] The subjects considered in the second absolute criterion (Vogel, 1996), do not appear in the following analysis. This is because those subjects were required to rate *all* of the negative chain criterion problems contrary to the literature's predictions. According to the present criterion, those subjects all fall into group 8, and are thus outwith discussion. On the other hand, subjects from the first absolute and its relaxation (in which negativity was held as the literature predicts and transitivity was relaxed) do participate here in groups 2 and 5.

rion, with no outliers.[10] These subjects' responses are classifiable in terms of the same basic reasoning strategies for those problems classified definitely as described for Group 1. Two of the subjects' definite responses were mainly those predicted by shortest path reasoning (8 problems) and the remainder by explicit link acceptance (a flavor of shortest path reasoning; 3 and 4 problems, respectively). One of these 2 subjects is #12, whose responses were detailed above. Moreover, there was a high rate of consistency between these subjects with respect to the problems they found classifiable in the way they did. Of the 4 remaining subjects in this group there was a more even distribution of correspondences to the three strategies. One of these subjects gave positive definite classification to 4 additional problems from which a definite reasoning pattern cannot be abstracted.

Group 3. There were 6 subjects in this group as well, with an average of only 5.2 definite answers (σ=3.3) (one subject was deemed an outlier in this group and excluded from further analysis, including the just-mentioned average: this subject did not give answers to 16 of the problems; 6 of those were among the inclusion criteria (which include this subject because inappropriate answers were not given) and 10 among the remaining 25 problems). One subject's (#75) definite responses were all predicted by explicit link acceptance. The remainder were split between explicit link acceptance, shortest path reasoning proper, most paths reasoning, and non-predicted reasoning. One of the latter 2 subjects was in the sentence-only condition and gave 2 answers which upon inspection were consistent with the polarity of the first path traceable as a chain of sentences, given the order presented.

Group 4. This group marks a transition into greater inconsistency. It contains 16 subjects, with an average of 11 problems each (σ=5.0) classified definitely, beyond the inclusion criteria. Among those, most paths reasoning was most frequently the best classification of responses (on average, 2.3 times per subject (σ=1.3)). Explicit link acceptance was the best predictor 1.1 times per subject (σ=0.8), and shortest path reasoning proper, 2.0 times (σ=1.3). Of the remaining problems classified definitely, an average of 2.9 (σ=2.4) could not be easily predicted by any current theory. There was an even distribution of subjects in this group between graphical and sentential conditions. The subjects with many problems answered 'strangely' definite tended to be in the sentence-only condition. Their responses, on inspection, coincided with the polarity first chain of sentences available, according to the order in which they were presented. In sum, this group also tended to skepticism, and most path reasoning was the best classification of definite responses.

Group 5. Although the average number of problems per subject given definite responses was 9.7 (σ=5.3), Three skeptics made 3 or fewer definite classifications (two were in the graph-only condition, and one in the sentence-only condition, with 10 of the 17 total subjects in this group in sentence only, and 7 in the graphical conditions). Of the remaining subjects, 1.2 (σ=1.3) was the average number of times/subject that explicit link acceptance was predictive; 2.2 (σ=1.6), shortest path reasoning, 1.8 (σ=1.2) most paths reasoning. There were 3.1 problems on average (σ=2.3), per subject which were not classifiable by any known theory of inheritance. Again, these tended to follow from subjects participating in the sentence-only condition, and were consistent with the polarity of the first path constructible between the queried classes, according to the order in which the sentences were listed. Again, in sum, we have support mainly for skepticism, and among definite (nonarbitrary) responses, for a combination of shortest path and most paths reasoning. Note that group 5 is the closest group within consideration to the subjects analyzed in § 2.1.2 in terms of relaxing the negativity criterion. That is, subjects in group 5 are much more likely than group 2 (say) to rate negative chains as negative. Examining the answers of subjects in this group in light of the possibility that they tended to consistently rate negative chains as negative yields 3.4 (σ=2.2) as the average number of times most paths reasoning was applicable.

2.2.4 Analysis

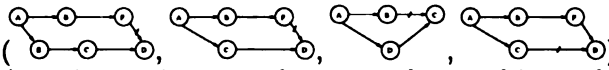
Now we examine the commonalities and variances among the groups. There were 8 groups altogether, but, as mentioned, after the level of relaxation represented by group 5 we did not find significant consistency. Taken together, groups 1, 2 and 3 had roughly the same number of subjects as each of groups 4 and 5 individually had; thus, we examine patterns which obtain among the regrouping g123, g4 and g5.[11] However, we also found roughly similar degrees of skepticism in groups 1, 2 and 3, and a significantly different (Tukey,[12] $p < .05$) degree of similarity between 4 and 5, so for other purposes we will also compare g123 with g45. Both of the regroupings are in simple supersets of the initial groupings described above.

First we mention systematicities across all groups. For example, there was not a significant difference in efficacy of explicit link acceptance in any of the groups. Also, there is not a significant difference in indeterminate ratings of 'classically' ambiguous problems

[10] Recall, such figures refer to problems other than those which defined inclusion.

[11] We hope the naming scheme is obvious — g4 is just group 4; g123 is the sum of groups 1, 2 and 3; and so on.

[12] This is the Tukey H. S. D. test, useful for *post hoc* analysis of differences in means; it is a more critical test than a t-test, for instance. Henceforth, in this section the name will be dropped and only the corresponding measure of significance given.

().

As an interesting example, among the 51 subjects who comprised the 5 groups, 77.6% rated the Nixon diamond indeterminate, 6.12% gave it a positive definite classification, and over double that, 16.3%, gave the problem a definite negative classification.[13] Both of these constancies serve to emphasize the generality of the consistency patterns observed in the original very small sets of absolutely consistent reasoners.

However, there remains evidence suggestive of partitions in reasoning strategies corresponding to the groupings that we defined by degrees of satisfaction of the inclusion criteria. First note that while a significant increase in 'arbitrary' responses does not occur through all 5 groups, it does obtain between g123 and both g4 ($p < .05$) and g5 ($p < .02$), as well as between g123 and g45 ($p < .01$). Also, there is a significant trend towards determinate responses between g123 and g45 ($p < .05$). Both of these facts should be anticipated by the very nature of the group definitions. For shortest path reasoning there is a very high mean of within subject applicability of the strategy for subjects in group 2. If explicit link acceptance is included as accepting the shortest path, then the difference in means for group 2 and each of the other 4 is significant (g1: $p < .05$; g3, $p < .01$; g4, $p < .05$; g5, $p < .05$). Group 2 seems to be best described then as shortest path reasoners. As for most paths reasoning, there is a significant difference in means between groups123 and both group4 ($p < .01$) and group5 ($p < .05$), as well as one between the combined g123 vs. g45 ($p < .01$), with a rise for the number of problems that could be classified by most paths reasoning for the more inconsistent groups. As the negative reasoners in Vogel (1996) also could be classified best as most paths reasoners this could point at a correlation between increasing negative chains rated negative and most paths reasoning.

The attentive reader will recall that the answers predicted by the various reasoning strategies agree in some cases with predictions made by systems that exist in the literature. Indeed, roughly the same patterns of between-group significance obtain for H90 as does for the shortest-path reasoner we've described (e.g. distinguishing group 2 from the others, and setting g123 as more in accord with the predictor than g4 or g5). However, these results follow almost exactly from the problems on which H90 agrees with shortest path reasoning. In fact, the shortest path reasoner was a strictly better predictor (by 50%), on the problems for which the two reasoners predicted different responses. The skeptical reasoner, different from H90 in allowing ambiguities to cascade and in demanding only on-path preemption, does in fact predict answers quite well for groups 1, 2 and 3, in comparison to H90, mainly because it is skeptical. We've already seen that subjects in those groups are more skeptical than the others. We've also claimed that the definite classifications of group 2 in particular is best predicted by shortest path reasoning. The cases in which skeptical reasoning out performed shortest path reasoning were also just those which generated indeterminate classifications.

[13] In comparison, the pilot study reported by Elio and Pelletier (1993) had 50% indeterminate response to the Nixon diamond, with the remainder split equally between positive and negative definite classifications. The between-subject analysis in the initial experiment of Hewson and Vogel (1994) yielded: 65% ind., 15% pos., 19% neg.

3 FINAL REMARKS

We have identified five reasoning strategies based on patterns of response within individual subjects classifications of the problems. Subjects responses were mainly skeptical, skeptical but accepting of explicit links, accepting of shortest paths generally, and/or accepting of most paths reasoning. The result is a family of reasoners which can, on the basis of evidence so far collected, be regarded as psychologically plausible. We have given details of the structure available in the data which supports these judgements. We found that the basically consistent subjects were roughly split between being very skeptical and less so (having found in the context of these experiments that people behaved mainly in accord with skepticism). Among the more skeptical, shortest path reasoning was the best predictor of definite responses. Among the less skeptical, and particularly among those who rate negative chains as definitely negative rather than as indeterminate as the literature expects, most paths reasoning was the best fitting predictor.

We emphasize that this has been an exercise in mining the data accumulated by Hewson and Vogel (1994) and Vogel (1995). While those experiments were designed with in mind the sorts of scrutiny we have engaged in here, we also investigated other issues which were apparent in the data by surprise. This constitutes a caveat with respect to accepted psychological methodology, and certainly future experiments we conduct will be obliged to investigate these issues directly from the outset. Nonetheless, the clarity of the results and the degree to which these conclusions are supported let us feel secure that they are not actually based on spurious significances.

This paper has reported within-subject analyses of the data accumulated during the initial experiments investigating the plausibility of inheritance reasoning as a model of human reasoning with defaults. Our ongoing work in this area includes replications and a battery of other experiments to clarify some of the issues that have been opened during analyses like the present one. For example, we are very interested in obtaining a clearer understanding of the role of mode of presentation. The current results hint at a more polarized response among those subjects who received the prob-

lems in a graphical condition than those who had only the sentential presentations of problems to work with, however, we do not yet understand how to interpret these facts, particularly since the same basic patterns of response that we mention here obtained in both the sentence-only condition and the graphical conditions. The exception is that group 2, which was mainly comprised of subjects in the sentence only condition, was the group that also best satisfied the predictions of shortest path reasoning. We lack evidence suggesting that these two facts are related, but it is a fact which begs further investigation.

Additionally, we are designing another style of experiment which involves a task following the classification of a problem and depending on the classification made. We hope that in such a context we can induce a greater proportion of definite classifications, since there will be a need to act on the conclusions which was lacking in the experiments discussed here. The differences in patterns of response, if any, will be rather interesting to learn. One of the problems with the design of the experiments which generated the data analyzed here is that the questionnaire was rather long, and the time required to deal with it seriously would certainly have encouraged frequent use of the psychologically 'easiest' of the available answers: indeterminate. Of course, we cannot enumerate here the full range of experimental questions that this work has raised in our minds. We hope, though, to have made clear that there is an extremely interesting set of issues to consider through further experimentation, and that the result will be informative to researchers who desire to build psychologically plausible formal models of default reasoning. We have offered preliminary suggestions for nonmonotonic reasoning strategies that people might be using in some circumstances, but our basic point is that plausibility judgements should not derive solely from a logician's introspection on examples but rather from observation of consistent rational behaviors that people actually exhibit.

Acknowledgements

Deep thanks to Robin Cooper, Claire Hewson, Jon Oberlander and Jeff Pelletier. Vogel is grateful for the Marshall Scholarship that let him study at the Centre for Cognitive Science in Edinburgh, and to the German SFB 340 for allowing him to work in Stuttgart.

References

Al-Asady, R. & Narayanan, A. (1993). More Notes on 'A Clash of Intuitions'. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pp. 682–7. Chambery, France.

Elio, R. & Pelletier, F. J. (1993). Human Benchmarks on AI's Benchmark Problems. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, pp. 406–411. June 18-21, 1993. Boulder, Colorado.

Evans, J. S. B. T. (1983). Linguistic Determinants of Bias in Conditional Reasoning. *Quarterly Journal of Experimental Psychology, 35*(A), 635–644.

Goldman, Alvin, I. (1986). *Epistemology and Cognition.* Cambridge: Harvard University Press.

Hewson, C. & Vogel, C. (1994). Psychological Evidence for Assumptions of Path-Based Inheritance Reasoning. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, pp. 409–14. Atlanta, Georgia.

Horty, J., Thomason, R., & Touretzky, D. (1990). A Skeptical Theory of Inheritance in Nonmonotonic Semantic Networks. *Artificial Intelligence, 42*(2-3), 311–48.

Kaufmann, H. & Goldstein, S. (1967). The Effects of Emotional Value of Conclusions upon Distortions in Syllogistic Reasoning. *Psychonomic Science, 7*, 367–8.

Niemelä, I. & Rintanen, J. (1994). On the Impact of Stratification on the Complexity of Nonmonotonic Reasoning. *Journal of Applied Non-Classical Logics, 4*(2), 141–79.

Rayner, K. & Pollatsek, A. (1989). *The Psychology of Reading.* Englewood Cliffs, NJ: Prentice Hall.

Selman, B. & Levesque, H. (1989). The Tractability of Path-Based Inheritance. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pp. 102–9. Detroit, Michigan.

Selman, B. & Levesque, H. (1993). The Complexity of Path-Based Defeasible Inheritance. *Artificial Intelligence, 62*, 303–39.

Touretzky, D. (1986). *The Mathematics of Inheritance Systems.* Los Altos, CA: Morgan Kaufman.

Touretzky, D., Horty, J., & Thomason, R. (1987). A Clash of Intuitions: The Current State of Non-Monotonic Inheritance Systems. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pp. 476–82. Milan, Italy.

Vogel, C. M. (1995). *Inheritance Reasoning: Psychological Plausibility, Proof Theory and Semantics.* Ph.D. thesis, Centre for Cognitive Science, University of Edinburgh.

Vogel, C. M. (1996). Human Reasoning with Negative Defaults. In Gabbay, D. & Ohlbach, H. J. (Eds.), *Practical Reasoning*, pp. 606–621. Lecture Notes in Artificial Intelligence 1085. Berlin: Springer Verlag. Proceedings of the *International Conference on Formal and Applied Practical Reasoning, FAPR'96.* Bonn, Germanny, June 1996.

Wason, P. C. & Johnson-Laird, P. N. (1972). *Psychology of Reasoning: Structure and Content.* Cambridge, Mass.: Harvard University Press.

Do Computers Need Common Sense?

Matthew L. Ginsberg
CIRL
1269 University of Oregon
Eugene, OR 97403-1269

Abstract

My aim in this paper is to make and defend three claims. First, it is incumbent on the knowledge representation and nonmonotonic communities to demonstrate that their ideas will eventually lead to improvements in the performance of implemented systems. Second, a reasonable working definition of "commonsense" reasoning is that it is the process of using polynomial techniques to convert a large instance of an NP-hard problem to a smaller instance on which search techniques can be applied effectively. And finally, it is a consequence of these first two claims that the most pressing problem facing the commonsense community is the identification of realistic problems and problem structures for which commonsense reductions are both necessary and effective.

1 INTRODUCTION

One might study formal aspects of knowledge representation for at least two reasons. In the first case, the elegance of the associated theories might itself be compelling; formal theories of reasoning might attract interest in the same way that number theory might enthrall a French jurist.

The other potential justification is more pragmatic. Knowledge representation, many of us have argued, will eventually be good for something. We remain in some disagreement with regard to what, and our critics (this author included) are becoming increasingly harsh over our failure to deliver the promised goods.

While the rest of artificial intelligence was similarly lacking, these criticisms could be deferred. Recently, however, the rest of AI has begun to pick up the pace: In areas as diverse as game playing, manufacturing scheduling, and natural language processing, results of practical importance are beginning to appear. In all of these cases, programs' successes have been a consequence not of insight and commonsense understanding, but of raw computational power. Commonsense is our human way of muddling through situations that are too complex for us to deal with exactly; the general impression from AI's successes is that machines may not need to do this.

At a fundamental level, however, this must be wrong: Interesting problems still appear to be NP-hard or worse, and the fact that useful approximating techniques have been found for narrow applications does not appear to mean that similar methods will work for broader-based projects. Machines may need common sense for the same reason we do: It is the only effective approach on problems of interesting size.

At the moment, however, the empirical evidence is against this. It is no longer appropriate for the knowledge representation and reasoning community to simply assume that commonsense reasoning is a necessity and proceed with business as usual. Instead, we need to develop a coherent theory of application areas in which nonmonotonic and commonsense capabilities will lead to practical payoffs.

Let me begin by discussing three specific focuses of AI research and potential applications: game playing, scheduling, and planning. I will then try to draw some common observations from these domains and suggest the impact these observations should have on future work.

2 GAME PLAYING

Machine game-playing programs are getting better. What's more, they are progressing not by imitating human methods for solving problems, but by exploit-

ing the vast computational power that is available on current platforms.

Consider chess. A human player will consider some tens (or perhaps hundreds) of positions when selecting a move in a particular position. A machine, on the other hand, will consider millions. The February 12, 1996 report by the Associated Press on the match between Kasparov and Deep Blue says in part, "Deep Blue [is] an IBM computer that has a calculating capacity of 200 million moves a second."

There have been attempts to produce computer chess players that play the game in a more human-like way, such as Wilkins' PARADISE system [30]. Unfortunately, the performance of these systems is at best modest. Simon draws similar conclusions from this example, saying, "It could be that, because of the radical differences between electronic devices and brains, programs designed to be efficient [intelligent programs] would be totally different in architecture and process from systems designed to simulate human thinking [26, p. 121–122]."[1] Other games are similar; in checkers [24] and Othello [21], for example, machine performance (now exceeding human in both domains) is a consequence primarily of raw speed.

More recently, games of imperfect information have begun to have a similar feel. Koller and Pfeffer suggest that it will be possible to use novel pruning techniques to solve poker exactly, presenting preliminary results for an 8-card deck [19].

Bridge is another game of imperfect information that has thus far eluded competent computer play. But here, too, the power of exhaustive methods is beginning to be felt. Smith and Nau remark that, "Some programs ... do decision making by repeatedly generating random hypotheses for what hands the opponents might have, and doing a full game tree-search for each hypothesized hand – but this approach is feasible only late in the game, after most of the tricks have been played [28]." I have made a similar suggestion that Monte Carlo methods can be effectively combined with exhaustive search [13].

The time needed to search the full game tree in bridge has fallen dramatically, due in part to increasing speed in hardware and in part to the discovery of new pruning methods [14]. A recently implemented system plays full hands of bridge in tournament-acceptable times using the method described in the previous paragraph; the performance of this system is far superior to any of its predecessors [10].

In all of these cases, Simon's speculation has proven correct. The strongest machine players do indeed bear little resemblance to their human counterparts. The only games for which this observation is not currently true appear to be backgammon (Tesauro's *TD-Gammon* program is neural-net based and appears, at least on a surface level, to have a human-like architecture [29]) and Go, for which no competent computer program exists at all.

3 SCHEDULING

On a different note, consider the domain of job-shop scheduling. In a problem of this class, there are a variety of tasks that need to be completed, each of which uses a specific type of machine and lasts for a given duration. Given that the number of machines of each type is limited and that some tasks must be finished before others can start, the goal is to produce a schedule that completes all of the tasks as quickly as possible.

There are many examples of problems of this type; typical is the construction of artifacts such as aircraft. Indeed, a large problem that is related to aircraft manufacture specifically has been made available over the web (http://www.NeoSoft.com/~benchmrx). This problem involves 575 tasks competing for 17 resources, with each task using multiple resources.

Work on automatically solving problems of this type has progressed through three distinct phases. In the first, expert systems technology was used to capture human expertise in finding effective solutions. Typical here is Fox's ISIS system [9].

In the second phase, more general search tools were used, but human intuition was still exploited to guide and focus the search. Typical here is Sadeh's work [31]; the search engines themselves exploit intuitive notions such as "bottlenecks" that are specific to job-shop scheduling problems.

Current research simply applies the fastest search tools available, typically combining both systematic and local search methods in a fairly generic way. The best known solutions on the benchmark problems mentioned above, for example, combine the systematic technique of limited discrepancy search [17] and a local optimization technique known as "schedule pack-

[1] It is not clear, however, how sanguine Simon is about accepting this conclusion. He says earlier, "It would be of enormous interest today to know what knowledge, how organized, would be required for a chess program to play at grandmaster level without needing to search more (100 branches?) than a human grandmaster [26, p. 103–104]." The assumption is made that it is at least *possible* to get machines to behave like people.

ing" [6].

In all cases, performance has improved as the search techniques have come to rely more on speed and less on human insight. Sadeh's methods outperform Fox's on academic problems involving 25 jobs using 5 machines; these problems themselves are no longer challenging for more modern methods [7]. Our earlier observations remain valid: Effective machine methods and effective human methods are virtually unrelated.

Other areas of AI are similar. There has been a great deal of recent progress in natural language processing, resulting in substantially greater accuracy in automated parsing of the large corpora that are used as case studies by this community. Almost all of this progress has been a consequence of the community's recent focus on purely statistical techniques [4].

4 PLANNING

Finally, consider generative planning. This area is somewhat different from the preceding two in that little practical progress appears to have been made, but the lessons to be learned are unchanged.

In a typical generative planning problem, one is given a situation in which a handful of blocks has been scattered on a table and the goal is to build a tower of some kind. The problem is either to construct a sequence of block movements that achieves the goal, or to construct one that achieves it in minimal time.

From a pragmatic point of view, these problems are best solved using simple state-space search. Start with the initial situation, and work forward. Simple pruning heuristics can be constructed that allow this method to quickly solve problems involving some 40 blocks.[2] Slaney and Thiebaux have improved the performance to hundreds of blocks [27] by ignoring the planning problem altogether, instead replacing it directly with the NP-complete problem that theoretically underlies it [16].

AI planning research, however, has focussed on a completely different approach. Actions are selected that achieve each of the goals, and an attempt is then made to sequence the actions in a way that avoids possible conflicts in which a block is placed on top of another that is about to be moved, and so on. The interest in this kind of approach appears to be a consequence of the fact that backward reasoning is a much more "humanlike" planning mechanism than is state-space search.

As usual, there is an intuition that there is some potential computational payoff to the backward reasoning, action-based approach. Perhaps by selecting an action to achieve a goal in general as opposed to in a specific situation, the resulting planner will be more robust against changes in the domain or other difficulties. In actuality, however, these arguments have yet to yield effective implemented systems – a generative planner constructed in this fashion is typically incapable of dealing with more than seven or eight blocks. Penberthy and Weld's UCPOP system [23], in fact, cannot reliably handle more than four without the inclusion of domain-specific control information.[3]

Interestingly enough, it appears that a continuum is emerging in the planning community, with pure "generative" planners at one end and state-based planners on the other. This work began with Blum and Furst's GRAPHPLAN system and their observation [3] that the performance of generative planners could be improved if one committed to specific times at which the selected actions would be taken. Kautz and Selman took this idea further, extrapolating away the actions as well and producing an axiomatization that simply constructed potential states from their predecessors [18]. The axiomatization was then passed to one of a variety of propositional theorem provers, and the resulting system proved still more effective than GRAPHPLAN, solving problems involving 17 blocks. The axiomatization itself, however, is strikingly similar to a propositional encoding of the associated state-space search problem; as such, the Kautz/Selman planner is itself outperformed by dedicated state-space engines.

The range of tools available here raises the question of which of these implementations can actually be considered "planners." Massey's and Slaney/Thiebaux's blocks-world planning programs, incapable of solving any problems in any other domain, are clearly not planners. UCPOP clearly is. Kautz and Selman's work, which contains no action descriptions and relies on the encoding being propositional instead of first-order, seems somehow intermediate with regard to this metric. The Kautz/Selman "planner" cannot be extended to either use a first-order axiomatization or to manipulate action descriptions without substantial computational cost.

In fact, the inference engine underlying the Kautz-Selman planner is WSAT [25], a fast tool for finding models of satisfiable propositional theories in which each clause consists of three randomly selected literals. WSAT's performance appears to degrade when it is ap-

[2] Bart Massey, personal communication.

[3] Will Harvey, personal communication; also cited elsewhere [12].

plied to realistic problems with more structure [7, 20]. McAllester and I have explained this by suggesting that:

> The most difficult random problems appear to be tangles of closely related individual variables while naturally occurring problems tend to be tangles of sequences of related variables. [15, p. 227]

WSAT and related algorithms appear to be strongest at solving problems involving "puzzle mode" reasoning (tangles of closed related variables) instead of "commonsense" reasoning (tangles of *sequences* of related variables). Problems in commonsense reasoning seem to involve lengthy partial analyses in which the branching factor is low.

As an example, imagine the search involved in buying a friend a birthday gift. Deciding what to purchase simplifies the remaining problem enormously because we are not tempted to do things like look for a fishing pole at a jewelry store. Binding one variable (the choice of gift) immediately restricts the choice for many other variables (the choice of store and thus travel route, etc.). This is not true in puzzle mode reasoning. In the zebra puzzle, for example, knowing that the Hungarian smokes Lucky Strikes gives you no direct information about the color of the house occupied by the Norwegian.

The reason WSAT can be made to function effectively as a planner is because Kautz and Selman have managed to find a propositional encoding of the blocks world that highlights its puzzle mode nature; this encoding has extrapolated away all of the domain's commonsense features. The fact that they have removed the actions is no accident – the actions were, in some sense, the last vestige of commonsense reasoning in the blocks world. By using an axiomatization that avoids the inclusion of actions and refers to the state space alone, Kautz and Selman retain only the puzzle mode aspects of the problem.

Kautz and Selman appear to argue that their WSAT encoding is still a planner because it is possible, generally in linear time, to recover the plan from the state sequence they produce. In their words, "Each pair of adjacent states corresponds to the (easy) problem of finding a partially-ordered plan of length 1." But we have argued that commonsense reasoning is different from puzzle mode work only *because* of the presence of an additional polynomial complexity at each node. Planning is hard not because the associated puzzle mode problems are computationally challenging, but because the automatic reduction from the original formulation to the puzzle mode problem is so difficult. Kautz and Selman admit explicitly that they are unable to automate this reduction process within their framework [18], and there are reasons to believe that this difficulty will not be easily surmounted, if at all [2].

Kautz and Selman have sidestepped the commonsense aspect of the planning problem. In doing so, it is not clear to me that their results embody a more compelling message than the fact that planners working in puzzle mode (i.e., using state-space search) will outperform planners that need to deal with more commonsense representations. A dedicated blocks-world planner such as Massey's or Slaney/Thiebaux's makes this point even more compellingly.

5 COMMONSENSE REASONING

What does all of this say about commonsense reasoning? The observations that we can draw from the preceding sections are completely uniform: Machines are far better than we are at puzzle mode reasoning, and we are far better at common sense. Furthermore, it often appears to be possible to encode problems involving common sense in a way that allows them to be solved using the machine's substantial puzzle-mode capabilities. Let me in fact take this observation one step further, and elevate it to the status of a definition:

Definition 5.1 Commonsense reasoning *is the technique whereby an inference problem is solved by reducing it (perhaps unsoundly) to a modified NP-hard problem on which exhaustive search is practical.*

My aim in presenting this definition is to say something sharp enough that instances of commonsense can be identified in a reasonably objective way, as opposed to by an appeal to an intuitive but otherwise unspecified standard. As with any attempt to define a previously unformalized notion, the justification for the definition should be that it does indeed manage to capture a wide range of examples of the phenomenon in question. Before moving forward, let me examine some instances of commonsense reasoning and see how they fit in this framework.

Consider the commonsense observation that the names of the individuals crossing the river are irrelevant in the missionaries and cannibals problem. Naming the missionaries Jimmy, Billy and Jerry (to pick three completely unmotivated choices) serves merely to increase the size of the search space without changing its structure in any interesting way. By recognizing that the names are not important to the solution of the problem, we reduce the problem to one that we can solve

using exhaustive search.

The mutilated checkerboard problem is similar; the commonsense observation that one should consider only the colors of the covered squares corresponds, as in the previous example, to a reformulation for which exhaustive search becomes practical.

Other examples of commonsense reasoning are nonmonotonic. Consider the recognition that the best way to get to the grocery store is to drive, assuming that there is still gas in the car, that it's still parked where you left it, that the supermarket hasn't moved either, and so on. Without these assumptions, the problem of getting one's groceries would simply be insolubly difficult. As I have suggested elsewhere, the purpose of nonmonotonic inference rules is to focus our attention on aspects of the problem that render it tractable [11]. McCarthy takes a similar view in his typology of uses for nonmonotonic reasoning, mentioning the use of nonmonotonicities as "rules of conjecture" [22].

Kautz and Selman's "compiling away" of the commonsense elements of blocks-world planning provides a final example. Our definition suggests that in this case, it is Kautz and Selman who are solving the commonsense aspects of the problem; their "planner" is solving the puzzle-mode kernel of the problem instead of the problem itself. This explains both the computational success of their method and the fact that they have not yet managed to automate the process by which a conventional description of a domain in terms of actions and fluents can be changed into one suitable for their planner. The fact that their planner is so incapable of dealing with the commonsense aspects of planning also explains the cognitive dissonance in calling it a "planner" at all.

Note, however, that Definition 5.1 views commonsense reasoning as fundamentally a heuristic activity, as opposed to an epistemological one. Although in keeping with my understanding of nonmonotonic reasoning [11] and with work such as Kautz and Selman's, this focus is very much at odds with the conventional view in the knowledge representation community. Nevertheless, I would argue that the definition works: Most if not all of what we refer to as common sense is covered, and little else is.

Given Definition 5.1, the work on scheduling and on game playing supports the view that automated commonsense reasoning is unnecessary because machines will simply deal with large NP-hard problems in their entirety. The commonsense portion of the reasoning exploits human insights regarding the nature of the search space. In scheduling, for example, progress in AI accelerated when we adopted the operations research community's idea that the branch points in the search space should not be the times at which various tasks are executed, but the selection of jobs that should be given priority when there is a resource conflict. In game playing, there is a long history of algorithmic developments, ranging from alpha-beta pruning to more recent work on singular extensions [1]. This last idea, for example, captures our commonsense intuition that forced moves should not contribute to the depth limit to which a machine searches. Alpha-beta pruning captures the fact that moves provably off the main line need not be considered.

In all of these cases, human insight is applied globally, transforming a problem in game playing, scheduling or planning into a modified form for which brute-force search is effective. The uniform success of this methodology supports the view that human and machine abilities simply differ. Perhaps we are meant to complement one another, with the humans reducing problems requiring common sense to problems that don't. Our job is to translate the problems into a form with which machines can cope effectively.

A more optimistic tack, however, and certainly the one best in keeping with AI's overall goals, is that machines will need common sense themselves. Such a need has been presupposed by the knowledge representation and reasoning community, and the time has come to justify that presupposition. The most compelling justification would involve the construction of algorithms that do just this, automatically modifying large and (apparently) intractable problems in ways that allowed them to be solved effectively.

This may be too hard, however. Only slightly less compelling would be the identification of problem types for which commonsense reductions were necessary. What structure can a problem have that will make an effective reduction possible while the problem itself becomes intractable for blind search methods? What evidence is there that such structure is present in naturally occurring problems? We need to be able to answer both questions here, finding problem features that can be exploited both in theory and practice.

This is clearly a problem with both a theoretical and an experimental component, but theory and experiment have diverged. Far too much theoretical work in this area foscusses simply on identifying structures with computational value, presenting no arguments whatsoever that these structures will or do occur in interesting problems. One need not look far in the literature on tractable deduction to find a variety of ex-

amples; indeed, the justification McAllester and I give for "crystallographic" problem structure [15] seems, in retrospect, far too flimsy.

Much of the methodology that has been proposed on the experimental side is similarly off target. Consider Cohen's "modeling, analysis and design" proposal for AI research in general, where the thrust of any particular piece of work is expected to be a theoretical or (more likely) experimental investigation into the performance of an implemented system as a parameter is varied continuously [5]. This "MAD" methodology (the acronym is Cohen's) completely ignores questions involving the theoretical identification of situations or problems for which classes of algorithms are well suited.

Other experimental work has focused almost exclusively on randomly generated problems that are designed specifically so as *not* to have the kind of structure that might support reasoning mechanisms other than puzzle-mode. This work has been an invaluable source of insight with regard to puzzle-mode reasoning itself, but to believe that it can lead to similar insights when applied to problems of common sense seems to me to miss the mark completely. As such, I am distressed to see recent work such as that of Etherington and Crawford [8] proposing to investigate experimentally the viability of nonmonotonic techniques in a random setting. If our goal is to identify structures for which commonsense reductions are a necessity, randomly generated problems are exactly the *wrong* place in which to look.

Overall, the problem of finding structures that support or require some kind of commonsense reasoning is initially a theoretical question as opposed to an experimental one. We need to find conditions under which some large instances of NP-complete problems can be made manageable by exploiting their structural properties. Having identified the structures, we then need to then provide experimental evidence that these structures actually appear in interesting and difficult problems. It is unfortunate that these sorts of questions are receiving so little attention from the knowledge representation and reasoning community: Not only are we the best placed to solve them, but we also have the most to gain from doing so.

Acknowledgement

This work has been supported by ARPA/Rome Labs under contracts F30602-91-C-0036 and F30602-93-C-00031. I would like to thank David Etherington and the members of CIRL for discussing these ideas with me.

References

[1] T. Anantharaman, M. S. Campbell, and F. Hsu. Singular extensions: Adding selectivity to brute-force searching. *Artificial Intelligence*, 43:99–109, 1990.

[2] T. Bedrax-Weiss, A. K. Jonsson, and M. L. Ginsberg. Unsolved problems in planning as constraint satisfaction. Unpublished manuscript, 1996.

[3] A. L. Blum and M. L. Furst. Fast planning through planning graph analysis. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1636–1642, 1995.

[4] E. Charniak. *Statistical Language Learning.* MIT Press, Cambridge, MA, 1993.

[5] P. R. Cohen. A survey of the Eighth National Conference of Artificial Intelligence: Pulling together or pulling apart? *AI Magazine*, 12(1):16–41, 1991.

[6] J. M. Crawford. An approach to resource constrained project scheduling. In *Artificial Intelligence and Manufacturing Research Workshop*, 1992.

[7] J. M. Crawford and A. B. Baker. Experimental results on the application of satisfiability algorithms to scheduling problems. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 1994.

[8] D. W. Etherington and J. M. Crawford. Toward efficient default reasoning. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1996.

[9] M. S. Fox. *Constraint-Directed Search: A Case Study of Job-Shop Scheduling.* Morgann Kaufmann, San Francisco, CA, 1987.

[10] M. L. Ginsberg. GIB and the May, 1996 Bridge World. *The Bridge World.* Submitted.

[11] M. L. Ginsberg. The computational value of nonmonotonic reasoning. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, Boston, MA, 1991.

[12] M. L. Ginsberg. Approximate planning. *Artificial Intelligence*, 76:89–123, 1995.

[13] M. L. Ginsberg. How computers will play bridge. *The Bridge World*, 1996.

[14] M. L. Ginsberg. Partition search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1996.

[15] M. L. Ginsberg and D. A. McAllester. GSAT and dynamic backtracking. In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning*, Bonn, Germany, 1994.

[16] N. Gupta and D. S. Nau. Complexity results for blocks-world planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 629–633, 1991.

[17] W. D. Harvey and M. L. Ginsberg. Limited discrepancy search. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 607–613, 1995.

[18] H. Kautz and B. Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1996.

[19] D. Koller and A. Pfeffer. Generating and solving imperfect information games. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1185–1192, 1995.

[20] K. Konolige. Easy to be hard: Difficult problems for greedy algorithms. In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning*, Bonn, Germany, 1994.

[21] K.-F. Lee and S. Mahajan. The development of a world class othello program. *Artificial Intelligence*, 43:21–36, 1990.

[22] J. McCarthy. Applications of circumscription to formalizing common sense knowledge. *Artificial Intelligence*, 28:89–116, 1986.

[23] J. S. Penberthy and D. S. Weld. UCPOP: A sound, complete partial order planner for ADL. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, pages 103–113, Boston, MA, 1992.

[24] J. Schaeffer, N. Treloar, P. Lu, and R. Lake. Man versus machine for the world checkers championship. *AI Magazine*, 14(2):28–35, 1993.

[25] B. Selman, H. A. Kautz, and B. Cohen. Local search strategies for satisfiability testing. In *Proceedings 1993 DIMACS Workshop on Maximum Clique, Graph Coloring, and Satisfiability*, 1993.

[26] H. A. Simon. Artificial intelligence: An empirical science. *Artificial Intelligence*, 77:95–127, 1995.

[27] J. K. Slaney and S. Thiebaux. Linear time near-optimal planning in the blocks world. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1996.

[28] S. J. Smith and D. S. Nau. Strategic planning for imperfect-information games. In *Proceedings of the AAAI Spring Symposium on Games: Planning and Learning*, Stanford, California, 1993.

[29] G. Tesauro. Temporal difference learning and TD-Gammon. *Commun. ACM*, 38(3):58–68, 1995.

[30] D. E. Wilkins. Using patterns and plans in chess. *Artificial Intelligence*, 14:165–203, 1980.

[31] Y. Xiong, N. Sadeh, and K. Sycara. Intelligent backtracking techniques for job shop scheduling. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, Boston, MA, 1992.

Actual Possibilities

Aaron Sloman
School of Computer Science
The University of Birmingham
Birmingham, B15 2TT, England
A.Sloman@cs.bham.ac.uk

Abstract

This is a philosophical 'position paper', starting from the observation that we have an intuitive grasp of a family of related concepts of "possibility", "causation" and "constraint" which we often use in thinking about complex mechanisms, and perhaps also in perceptual processes, which according to Gibson are primarily concerned with detecting positive and negative affordances, such as support, obstruction, graspability, etc. We are able to talk about, think about, and perceive possibilities, such as possible shapes, possible pressures, possible motions, and also risks, opportunities and dangers. We can also think about constraints linking such possibilities. If such abilities are useful to us (and perhaps other animals) they may be equally useful to intelligent artefacts. All this bears on a collection of different more technical topics, including modal logic, constraint analysis, qualitative reasoning, naive physics, the analysis of functionality, and the modelling design processes. The paper suggests that our ability to use knowledge about "de-re" modality is more primitive than the ability to use "de-dicto" modalities, in which modal operators are applied to sentences. The paper explores these ideas, links them to notions of "causation" and "machine", suggests that they are applicable to virtual or abstract machines as well as physical machines. Some conclusions are drawn regarding the nature of mind and consciousness.

1 Introduction: possibilities everywhere

This paper is a conceptual exploration and does not pretend to offer any formal results. It suggests some new ways of looking at old problems which may be relevant both to AI researchers designing robots or systems that need to reason about complex mechanisms, and to cognitive scientists investigating human and animal capabilities.

The orientation is primarily ontological, i.e. concerned with the sorts of things whose existence we (or animals or future robots) take for granted both in our perception and in our problem solving and reasoning. Exactly how we (and other animals or robots) represent and reason about the sort of ontology described here is a topic for future research, which will have to formalise the ideas and embody them in useful mechanisms. Much vagueness is inevitable at this stage and not all questions raised will be answered nor all the conjectures justified.

The key idea is that the world contains not only objects with properties and relationships, but also *possibilities* and *links* between or *constraints* on possibilities. The latter are the causal powers of objects. Some of the possibilities are things we like or dislike, hope for or fear, and we label them risks, dangers, opportunities, prospects, temptations and so on. Others, described more neutrally, are of interest because of the role they play in explanatory and predictive theories, for instance, possible velocities or accelerations of an object, possible voltages across a conductor, possible pressures in a gas. Often the functional role of an object is concerned with possible states or processes that object can enter into, which, in turn, will change the possibilities for other things. An old example from planning is an action whose effects influence the preconditions for another action.

1.1 Perception of possibilities

In Gibson's theory of perception as the acquisition of affordances (Gibson 1986), the key idea is that for an organism merely to be given information about the structures of objects in the environment is not sufficient for its needs. Gibson's claim is that many organisms have evolved the ability to perceive what he calls "affordances", which involve sets of possibilities and constraints on possibilities. For instance, a surface affords support for the animal, an opening in a wall affords passage, a berry affords picking and eating. He makes a second claim that these are not inferences made by central cognitive mechanisms on the basis of structural information delivered by the visual system. Rather the detection and representation of affordances happens deep within the perceptual mechanisms themselves. The first claim may be true even if the second one is not.

I tried to elaborate both claims in Sloman (1989). Compare Pryor (1996) for related ideas regarding "reference features" used by reactive planners. I'll return below to questions about the nature of the various types of affordances that animals can perceive and use in their actions, and about how their knowledge is represented and manipulated.

1.2 Possibilities, causation and change

Some features of an object are permanent while other features and relationships can change[1]. Objects may change their size, their colour, their temperature, their orientation, their distance apart, their containment relationships and so on. The properties and relationships of an object that are capable of changing may be thought of as "selections" from ranges of possible properties and relationships. (I don't mean that selections are conscious or deliberate). When things change different selections are made.

Some of these sets involve global features, such as size, weight, or orientation. Some involve relations of the whole object to other things, e.g. containment, distance and direction from other objects, motion towards something, attraction between objects, and many more.

Some of the sets of possibilities inherent in an object are analysable in terms of possibilities involving *parts* of the object. For example whether an object is striped or not depends on whether parts have the same colour. The shape of an object depends on spatial relations between parts of the object. The degree of compression of an object depends on both spatial relations between and forces between parts of the object. The degree of flexibility involves possible changes in spatial relations between parts.

The sets of possibilities inherent in an object help to define the nature of the object, or at least what it is about the object that needs to concern an animal or robot that interacts with it, or a designer who embeds it in a larger configuration. Some of these possibilities will be perceivable simply on the basis of sensory qualities (e.g. those that depend on shape, spatial relations, observed motion, resistance to pressure, tactile qualities, etc.) while others require inference or learnt associations, e.g. edibility, risk of being stung, what an agent can see, etc.

1.3 Constraints and causal links

Selections from different sets of possibilities cannot all be made independently: there are *laws* or *constraints* linking different ranges of possibilities. These links, which may depend on context as well as the nature of the object, are themselves higher level properties, and insofar as the links or constraints can change they too may be selections from more abstract (higher order) ranges of possibilities, i.e. possible constraints between possibilities.

Certain properties of an object can be directly altered from outside, for instance, its temperature, or the voltage across it. This involves a selection from one of the ranges of possibilities. Typically this will *cause* the selections from one or more other sets of possibilities to change. For example, changing the temperature may cause the length to change. Exactly how changes in one set cause changes in another depends on the object (and possibly its environment). We can think of the object as a "transducer" linking sets of possibilities.

1.4 An example: electrical resistance.

Many physical properties of objects involve transduction relations between possibilities, for instance electrical resistance of a conductor.

This can be understood without knowing about deep explanations of electrical phenomena. It suffices to know that some objects are able to conduct electricity, that flow of an electric current can be produced by applying a voltage across the conductor, that both voltage and current can be measured and that there is always a fixed relationship between the two states.

More precisely, talking about resistance of a conductor presupposes that there is a set P1 of possible voltages across the conductor, a set P2 of possible currents in the conductor, and that the conductor has a property R that limits the selections from the two sets of possibilities so that the ratio between voltage and current is constant.

So a piece of wire can be seen as a transducer from one range of possibilities (possible voltages, P1) to another range of possibilities (possible currents, P2) with the ability to constrain the relationship between inputs (voltages) and outputs (currents). That ability is its electrical resistance.

Although we measure both the voltage and the current using numbers, the numbers are multipliers for something physical, unit voltage and unit current, which depend on the system of measurement in use. So resistance is not just a numerical ratio, though it can be represented by one.

Thus construed, resistance is an abstract property of conductors. Abstract properties typically are *implemented* in other properties. Some time after learning about resistance, physicists learnt more about the architecture of matter, including underlying mechanisms and properties in terms of which resistance (and the ability to conduct electricity) is "implemented". I shall not discuss the philosophical question whether the resistance IS those other things or not. Certainly physicists knew about resistance as characterised here before learning about electrons, quantum mechanics, etc. So there is at least an epistemological distinction between resistance and the underlying properties, whether there is an ontological difference or not.

There probably still are many engineers who know nothing about the underlying physics, and manage with concepts of properties defined solely in terms of ranges of possibilities and constraints linking those ranges of possibilities, like

[1] Degrees of permanence are discussed later in connection with remoteness of possibilities

the constraint expressed in the equation **V = RC**. For engineering purposes, that sort of "surface" knowledge is often more useful than deeper knowledge about the underlying physical implementation. However, the deeper knowledge may be required for the task of designing new conducting materials. In that case we have to understand how features of the architecture of matter are relevant to the possibility of electrical currents.

2 Other physical properties

Electrical resistance is just one among many examples of properties linking sets of possibilities. A conducting piece of wire may also have a modulus of elasticity, which is a physical property associated with two ranges of possibilities, namely possible tensile forces that can be applied to the ends of the wire and possible changes in length of the wire. As with resistance, the wire links particular possibilities in the first range with selections in the second range, and within a sub-range of each set the ratio between the measures of the input possibility and output possibility is fixed. Beyond that subrange inelastic deformation occurs.

In some conductors changes in temperature produce changes in resistance. Here we have a range of possible temperatures P3, linked to a range of possible resistances P4, where each resistance is itself a particular link between sets P1 and P2, i.e. possible voltages and possible currents through the conductor. So some physical properties involve second-order possibility transducers. The extent to which temperature changes elasticity is another second-order link.

Some conducting devices are designed so that turning a knob or moving a slider changes the resistance between two terminals, by changing the length of wire used between the terminals. Here the set of possible orientations of the knob is mechanically linked to the set of possible lengths of wire through which the current flows, and this is linked to a set of possible resistances. Thus the relation between knob position and resistance is another second order link.

Not all causal links between possibilities involve linear relationships. In an amplifier's volume control the link between possible orientations and possible resistances might be logarithmic. In a gas held at a fixed temperature changes of pressure produce changes in volume and the first is inversely proportional to the second, at least within certain pressure and temperature ranges. However, a monotonic non-linear relation can often be transformed into a linear one by remapping either P1 or P2. (E.g. replace volume with its inverse.) But there are more interesting exceptions.

2.1 More complex interactions

Many of the possibility-linking properties are expressible in terms of mathematical functions linking numerical values, but we also have names for linkages between qualitatively different possibilities. For example a vase admits a variety of possible interactions with other objects: it could be struck by a fast moving bullet, hit by a slower moving sledge hammer, squashed between two plates moving together very slowly, or dropped on a hard floor. Call the set of possible damage-causing interactions P1. The vase is also capable of changing into a very wide variety of states in which its wholeness is lost: it may break into two pieces in many different ways, or it can break into large numbers of fragments, also in very many different ways. Call that set of possible states P2. Depending on the material and its shape, it will have a causal property linking the occurrence of any element of P1 to a selection from P2.

We have various names for such properties, including "fragile", "brittle", "breakable" and "delicate". (The latter is sometimes used to describe the type of *appearance* that is associated with the causal property of being delicate, i.e. easily broken because components are very slender.) If the vase is made of glass or china it will probably be both brittle and fragile. If the material is very thin it may also be delicate. It is interesting that we also use these words to describe more abstract properties of non-physical objects, for example a fragile personality, delicate health, fragile ecosystem, etc. because of strong analogies between different sorts of causal properties linking quite different sets of possibilities.

These are all standard examples of *dispositional* concepts, concerned with "what would happen if...", i.e. all referring to linkages between sets of possibilities. Many engineers, technicians and craftsmen have to become familiar with a wide range of such properties, including knowing how to recognise when they are present, knowing how to make use of them when designing, making or fixing things, knowing how to recognise the conditions that will trigger the linkages, knowing how to prevent such conditions arising, e.g. by choice of storage conditions or packing materials.

2.2 Combinations of inputs are sometimes relevant

So far I have written as if all possibility transduction involves a link between two sets of possibilities P1 and P2. However, there are many cases in physics where one measurable depends on two or more others. E.g. in every case where I talked about a second-order transducer, it is possible instead to talk of a first-order transducer with two inputs. E.g. current in a wire can depend on both voltage and temperature. We just don't happen to have a name for this relationship between three sets of possibilities, though perhaps we could find one useful.

Many of the properties linking sets of possibilities depend on a combination of factors such as the material used, the shape or structure of the object, and the environment. For instance a vase made of lead will behave very differently from a vase made of glass when dropped. Moreover the properties may change when the vase is at a very high or very low temperature, or immersed in a liquid, e.g. treacle. Thus in general the selection from the set P2 can depend not just on a selection from one other set P1, but on a host of selections from other sets, P1a, P1b, P1c, etc. some in-

volving internal states of the object and others relationships with or states of the environment. Full replication of human abilities to use knowledge about objects in deciding what to do, making plans, controlling actions, would have to take account of all these different cases.

2.3 The input and output sets may be hard to characterise

The sets of possibilities linked by a causal property may be very hard to characterise in a formal and general way. We have an intuitive grasp of the range of possibilities P1 that would trigger a manifestation of fragility in a vase but would find it difficult to list necessary and sufficient conditions. Similarly the set of possible outcomes P2 associated with notions like fragility or brittleness can be very large and varied, and hard to characterise.

Nevertheless people can learn about both sets as well as learning how to detect objects which link P1 and P2. That raises interesting questions about how the learning is done and the form in which the knowledge is stored and used. It seems that the expertise of a craftsman or engineer can include the ability to distinguish far more different sorts of cases than we have words for in our language. For instance the word "fragile" can describe both a vase and a spider's web, yet different classes of input and output possibilities are relevant to their fragility.

2.4 The causal links may have different forms

The knowledge we have concerning the causal links between P1 and P2 may be very different in form, depending on the case. For example, knowing the resistance of a conductor involves knowing a very precise relationship between possible voltages and possible currents. The mapping between members of P1 and P2 is one to one and easily formalised. In the case of a fragile and brittle vase all we know is that each item in P1 will realise *one* of the possibilities in P2.

Common sense knowledge includes the ability to make coarse-grained predictions, for instance about the difference between dropping a vase and squashing it between two sheets of metal, or the difference between striking it on the side with a hammer and striking it with a downward blow. An expert may, as a result of years of experience, learn more about the relationship between members of P1 and members of P2, and on that basis can at least partially control the manner in which something breaks, for instance the occupant of a house breaking a window in such a way as to make it look like the work of a burglar doing it from outside, or a demolition expert using explosives to bring down a large disused chimney safely.

2.5 Discrete outcome devices

Gambling devices, such as a roulette wheel or a device in which a descending ball bounces against pins until it falls into one of a set of slots also allow only limited prediction. The difference is that, unlike a fragile vase, these devices are constructed so as to have a small finite set of end states (so that there's a well defined set on which to place bets). It is a remarkable fact about such devices that although particular outcomes are unpredictable their statistical properties are predictable: they have well defined probability distributions and these can be adjusted, e.g. with hidden magnets and the like.

So here we have a set of initial possibilities P1 and outcomes P2 where we cannot find specific links between selections from P1 and P2, whereas we do find a very strong link between sequences of selections from P1 and global numerical properties of the resulting selections from P2: hence the concept of a *probability distribution* as a property of an object. There are many philosophical puzzles about the concept of probability but I shall not go into them here. (Popper's notion of "propensity" is relevant.)

2.6 Causal determinism

In the case of a shattered vase we do not have the vocabulary to describe all the possible initial events nor the possible outcomes in precise detail. In the case of precisely engineered gambling devices we may have the vocabulary, but we still have no way of predicting precisely the outcome of an interaction, even if we use very precisely engineered machinery to replicate initial conditions (which is harder when we have to manufacture multiple vases under identical conditions).

A causal determinist will argue that in all these cases there is a precise and rigid linkage between initial and final states, but it is merely our knowledge that is incomplete: knowledge of the fine details of the initial conditions and knowledge of all the physical laws involved. The assumption is that if our knowledge were complete, predictions could be precise. That assumption is often taken to be a feature of classical physics.

Where chaotic phenomena are involved only *infinite* precision in the initial states could uniquely determine future outcomes. Clearly we cannot, even in principle, obtain infinitely precise measures of physical states (especially if those states can vary continuously, in which case most of the states could not be described using any finite description). Whether the actual states of physical systems could have infinite precision is not clear. It is normally taken to be an assumption of classical physics that they do, but since no classical physicist ever had any reason to believe that infinitely precise measurements or descriptions were possible, it is not clear how they could have believed that the states themselves were infinitely precise.

This opens up a new line of thought: even if the *laws* of physics are totally determinate, e.g. expressed in simple differential equations, it might still be the case that the *states* and *boundary conditions* cannot be infinitely precise. In that case systems with non-linear feedback such as produces chaos might be intrinsically unpredictable even

within classical physics.

That makes it even more remarkable that when the geometry of a chaotic gambling device forces a finite set of possible outcomes the long run frequencies should be determinate. This line of thought reduces some of the differences between classical physics and quantum physics: both can involve non-deterministic selections among finite sets of possibilities where the selections obey only statistical laws. The difference is that in these devices the classical possibilities are constrained by the geometry. This may not hold for all classical processes, e.g. the breaking of a vase may be unlike the process on a roulette wheel.

We've seen that engines that use no principles of quantum mechanics involve linked networks of possibilities. Any particular state of such an engine, or any particular extended piece of behaviour over a time interval is but one among many possibilities inherent in the design. All those possible voltages, currents, rotations, velocities, forces, etc. are real possibilities in the sense that the configuration of the machine allows them to occur, while other combinations of possibilities occurring in particular spatio-temporal relations are ruled out by the configuration. We'll look at different sorts of dynamics later.

As explained above, even when the details are unpredictable we may still have useful qualitative knowledge, which can guide our decisions regarding how to transport objects, how to predict types of breakage in a vase, how to make a broken window suggest a burglary, etc.

3 Types of knowledge and contexts of use

When a craftsman makes an object, some of the properties of the material in an object are particularly important during the process of construction and manufacture, others most important in the finished product.

E.g. the fact that wood can be cut, planed, sanded, etc. is important when an article is being made, whereas in the finished article hardness, rigidity, and low thermal conductivity may be more important.

Similarly some aspects of the shape of a building may be important during the process of construction, for instance making a partially completed structure stable, thereby reducing the need for scaffolding.

A particularly important sort of knowledge that the craftsman uses is knowledge of how to combine or shape materials so as to constrain the causal linkages. This may include grasping ways in which causal powers of some parts of an object combine with or constrain causal powers of other parts to produce properties of the whole. The global shape of an object, for example, involves various parts interacting with other parts to produce not only global geometric and topological features, but also global causal powers.

For instance, some shapes will make an object less liable to break under stress. Some shapes, such as boxes and bowls, will enable an object to hold other objects in place, which might otherwise roll around e.g. while being transported in a bumpy vehicle. Some shapes will contain a liquid. Some, but not all, will do both. I'll return to the role of knowledge in design later.

3.1 What do other animals know?

It is an interesting question to what extent animals that build nests or use tools or select where to walk or dig, or which branch to leap onto, have some grasp of the causal powers of different materials and of different structures. Do nest building birds have any grasp of the difference between ways of assembling materials that will produce a rigid structure and those that will not?

If Gibson is right about perception and affordances, then perhaps the magpie that uses its beak to insert a twig (which may be tens of centimetres in length) into its partly built nest, selects the location and direction of movement at least partly on the basis of perceiving the affordances for insertion and relative motion. The twig is held roughly at right angles to the beak, and even flying back to the nest with it, avoiding collision with branches on the way, is no mean achievement.

Insertion then requires first moving forward to one side of the insertion point (e.g. to the right of it), then inserting the end of the twig with a sideways motion of the head (e.g. to the left), and then repeatedly releasing the twig and grasping it nearer the free end and pushing it deeper into the nest with a sideways motion. Because of the intrinsic variability of shapes and sizes of twigs, partly built nests, and the configuration of branches in the supporting tree, no rigidly predetermined sequence of movements could achieve the task. So it looks as if the magpie needs some understanding of what it is doing.

Whether it also understands the consequences for the *long term* properties of the nest is doubtful. But even the ability to assemble the nest seems to involve using knowledge and abilities in a way that goes beyond what we currently know how to program into robots. This includes (a) the ability to perceive the affordances in the environment, (b) the ability to use the affordances to select actions (including selecting items for the nest, choosing route details while flying back without crashing into branches, deciding where to insert new bits into the nest) and (c) the ability to perform the actions, e.g. controlling fine details of the movement using visual and tactile feedback of changing affordances. In humans studies of how different sorts of brain damage can limit sorts of sub-skills may give clues as to which abilities to perceive, understand and use possibilities are used for different tasks.

4 Possibility transducers in virtual machines

So far all my examples have involved physical objects with physical properties which link and constrain sets of physical possibilities. Interactions with other information process-

ing agents (not all of which are friendly) make new classes of possibilities and possibility transduction important. For example it may be important to think about possible items of information another agent can access (e.g. what it might see or hear), what information it might have gained previously, and how it could use different items of information, e.g. in taking decisions or making predictions. Our perception of faces uses mechanisms designed not only to discriminate and recognize individuals but also to detect a variety of types of *internal* states. These must have evolved together with mechanisms for displaying those states. Similar capabilities are found, though in simpler forms, in many other animals, though it is very hard to tell exactly which possibilities they detect and reason about.

Machines can also have information processing capabilities. In computer science and software engineering, it is now commonplace to think about abstract non-physical entities as having causal powers.

For instance, a word processor includes characters, words, sentences, paragraphs, lines, pages, page numbers, and other abstractions. It also allows a host of possible changes in any configuration of a document, for instance, inserting, deleting moving, changing font sizes, changing line spacing, etc. Each such change has potential consequences, such as altering the length of a line, changing line breaks, changing the contents of a page, changing page numbering, triggering a spelling checker and so on. How a particular change affects the configuration may be controlled by other aspects of the state of the program that can be changed by different sorts of actions. For instance setting the line length or the page size will alter the effects of inserting a word. So here too there are higher order possibility transducers. But most of the possibilities are not physical.

Although the wordprocessor may display part of its current state in a physical image on the screen, the program primarily constructs and manipulates *abstract* structures. They are datastructures in a virtual machine, like the structures in operating systems, compilers, theorem provers, planners, plant control systems, office information systems, etc. Some of these will be entirely concerned with processes in virtual machines. Others are part of a larger structure that includes physical states and processes of complex systems linked to the software.

One of the features characteristic of such systems is that typically the changes that occur are not continuous changes in the values of a set of numerical variables. In general there are discrete changes in more or less complex structures and many of the events involve changes in complexity, or structural changes such as switching two sub-trees in a tree. A formal characterisation of such a set of possibilities would therefore typically require something more like a grammar defining a class of possible structures than a fixed list of variables whose values can change. I.e. the dynamics of such structures cannot easily be construed in terms of motion in a high dimensional vector space.

Of course, the underlying *implementation machine* will typically use a large boolean state vector: but that's normally irrelevant to understanding the design or principles of behaviour of a complex piece of software, just as the precise configuration of atoms in a car may be irrelevant both to the driver and the car mechanic, since both operate best at higher levels of abstraction. However, some of the high level behaviour of the car may be best characterised in terms of equations linking a fixed set of numerical variables, e.g. speed, acceleration, torque, fuel consumption, air resistance, coolant temperature, and many more.

Describing and explaining changes in a computer will normally require quite different sorts of mathematics.

To illustrate: understanding how to use a word processor or compiler or operating system involves acquiring a grasp of the ontology used by the software. That in turn involves not only learning which objects and configurations of objects can exist but also grasping the additional possibilities and constraints on or links between possibilities inherent in various configurations. In a software system, like a word processor, these will constantly be changing during use, just as the physical affordances constantly change during the building of a nest or some other physical structure. Expert users of software systems have to develop the ability to perceive, represent, and make use of these affordances, including the high level affordances.

A software design environment is one in which the affordances involve very high orders in multiple domains. For example there is typically some sort of editor which has its own affordances for text construction and manipulation, along with the file management system. Then there may be one or more compilers which afford transformations of the text being constructed so as to produce new software systems with their own affordances. The process of editing not only changes the structures and the affordances in the source text configuration but also the affordances for the compilation process and structures and affordances in the resulting software. Moreover as the program text develops not only are there new possibilities for textual changes there are also new possibilities at the level of the program design, for instance, introduction of a macro provides opportunities to simplify syntax both in existing text and in new text, and development of a procedure changes possibilities for use of that procedure as a subroutine in future procedures. If the development environment involves an interpreter or an incremental compiler and the editor is part of the same software system, as in many AI development environments, all these possibilities are linked in very intricate ways, which can be hard to learn, but once learnt can speed up development of sophisticated software considerably.

But what exactly does the user learn? How is the information about all these abstract and rapidly changing affordances represented? How is that representation used both in the process of high level design and the relatively low level choice of text manipulation operations while coding, running tests, inspecting test output, etc.?

These examples show that virtual machines in software

systems, like vases, admit of complex and varied sets of possibilities. However, like gambling devices, the underlying machinery is designed to allow only discrete sets of possibilities, and *unlike* both breaking vases and gambling devices the machinery constrains the dynamics to be totally predictable and repeatable (provided that the external environment and initial conditions do not change). It may be that these features are essential for reliable long term memory stores and intricate intelligent behaviour. Some people think the digital nature of such systems is a disadvantage, compared with neural nets allowing continuous variation of weights or excitation levels.

5 When are causal links mathematically expressible?

Both the electrical resistance of a wire and the fragility of the vase are features of the causal power of the object to "transduce" elements from one set of possibilities to another. In the former case we can express the relationship in a simple and precise mathematical formula, whereas the second case is far less rigid and precise and we do not have an appropriate mathematical representation. E.g. we have no set of mathematical descriptions of either range of possibilities (P1 and P2), even though we have an intuitive grasp of both since (a) we learn which forms of treatment to avoid so as to protect the vase and (b) we know which sorts of behaviours would be surprising and which would not, if the vase is struck, or dropped, or squeezed hard, etc. How our brains represent those classes and whether similar representations could work on computers is an interesting research question. Even if we had a precise way of characterising the two ranges of possibilities it is not clear that we could specify which element of the second range would be caused by an occurrence in the first range.

There are other cases where the causal link cannot be expressed as a simple mathematical formula linking numerical values. Computing systems are full of examples. For example, the behaviour of an interpreter for a simple programming language may be best expressed as a multi-branch conditional expression embedded in a looping construct of some sort. Here the range of input possibilities and the range of output possibilities may be expressible using languages with well defined grammars, and the transduction capability can be formally expressed in terms of rules for transforming input "sentences" into output "sentences".

In more complex cases the computer also stores records of what it has done, and these records can change future behaviour, in arbitrarily complex ways, for example if some inputs are programs that are compiled into internal stored instructions that can be invoked by later inputs. Organisms that learn also keep changing the affordances that some components offer to others.

From our present viewpoint these are just special cases of the general fact that as objects realise their possibility transducing potential they can change their future powers, for instance, by wearing out, becoming more flexible, storing information, losing energy, gaining energy, hysteresis, and so on.

6 Combining linked sets of possibilities

So far I have described, albeit in a sketchy and shallow way, some cases where we naturally see an object as having a collection of properties, where those properties are possibility transducers. Further work is needed to classify different cases, e.g. according to the structure of the set of possibilities, and according to the form of the relationships constraining possibilities.

One of the uses of a grasp of possibilities is in thinking about designs for new objects. This requires understanding the implications of combining objects into more complex objects. There are many instances of this in everyday life, some of them very simple, others more complex. Simple examples involve mere agglomerations with cumulative influences.

For example, a sheet laid on the ground has many possible motions, some of which will be produced by gusts of wind of various types. By placing a stone on each corner we can transform the links between possible gusts and possible motions, essentially by reducing the set of gusts that will result in the sheet blowing away: only the stronger gusts will produce that effect. By adding stones we can reduce the set even further. There are many examples that are very similar in form to this. If I stand against a wall with shelving containing books or other articles there is a range of postures I can take up that will enable me to reach objects on the shelves. If the shelving is high, many objects will be out of reach. By placing a block of wood on the floor so that I can stand on it, I can extend the range of possibilities. By adding further blocks, stacked vertically I can continue to extend the range of possibilities (though the blocks may also eliminate a subset by obstructing access to objects on lower shelves).

Combining objects into more complex wholes does not always have an additive effect on sets of possibilities. An interesting example is the case of two identical electrically conducting wires W1 and W2, which can either be joined at one end of each to make a new conductor of double the length of the original (joined in series) or joined at both ends to make a conductor of the same length (joined in parallel). Each piece of wire initially has a collection of properties, each linking sets of possibilities. But they are separate sets: possible currents in W1 are different things from possible currents in W2. Normally, the properties of W1 do not constrain the possibilities associated with W2, only the possibilities associated with W1. However if wires W1 and W2 are joined in series their currents are forced to be the same (at least in standard conditions). Similarly if their ends are joined in parallel, the voltages across them are constrained to be the same. Moreover in those two cases there are new relationships linking possible voltages

to possible currents: i.e. the combination has a different electrical resistance from either of its components, and the resistance will depend on the form of combination. More complex interactions occur when two conductors are adjacent and the current is changing in one, or one is moving relative to the other. These are facts that can be used in the design of dynamos and electric motors.

6.1 Towards a theory of design?

All this suggests a further research topic: investigation of forms of combination of objects into more complex objects and the effects of different sorts of combinations on the possibility-transducing powers of the new structures. Particular subsets of cases are to be found in books on circuit design, and the design of various sorts of mechanical structures, e.g. linkages which allow parts to move relative to one another but with constraints on the possibilities. Car boot lids and engine bonnets are familiar examples.

Many physical mechanisms, including both mechanical devices and electronic machines, both digital and analog, consist of collections of physical objects each of which has properties that associate different ranges of properties and limit combinations of possibilities from those ranges. We have discovered how to make the "output possibilities" of one physical object the "input possibilities" for another, and to construct networks of such such possibility transducers in such a way that the complete configuration does something useful, which might not happen naturally.

That is the task of a designer, and there are many very sophisticated designers, assembling fragments of various kinds into larger structures, including cars, gas turbines, aeroplanes, houses, football teams, commercial organisations and software systems. Walls and doors of houses link and limit very complex networks of possibilities: including transmission of sounds, of heat, of temperature, of people, of furniture, and, with the help of glue and nails, also constrain motion of pictures, wallpaper, etc. They also link and limit sets of possible psychological states of occupants.

In an old fashioned clock all the possibilities involve clearly visible state changes, such as rotation of wheels, downward movements of weights, oscillation of pendulums and ratchets, etc. In an electronic clock with a digital display the collections of interlinked possibilities are of a very different kind, and the laws associated with the various possibility transducers are very different, involving discrete rather than continuous changes, for example. But the general principle is the same in both cases: a complex machine is created from simple components by linking them in such a way that the associated possibilities are also linked, and thus we get large scale constrained behaviour that is useful to us, and emergent capabilities such as telling the time and providing shelter.

It seems that some people and some animals can only grasp the *perceivable* possibilities (Gibson's affordances) and can understand a complex design only insofar as it links together perceivable structures. However, good human designers can also think about more abstract cases including both unobservable physical structures (chemical structures, submicroscopic digital circuits) and also abstract structures in virtual machines. Are very different forms of representation used in these two sorts of cases?

7 Damage vs change: more and less remote possibilities.

We have talked about complex systems where some combinations of the possibilities associated with components are permitted and others ruled out, at least while the configuration is preserved. Breaking a machine in some way destroys the configuration and allows some new possibilities to come into existence, and perhaps removes others. Can we make a principled distinction between changes that should be regarded as damage and changes that preserve the integrity of the system?

This depends on whether we can identify certain sorts of objects as having some sort of self-maintaining capability. Holland (1995) discusses many cases, including individual organisms, cities and social systems. For now it is not important for us whether there is a well defined set of cases. There may be a range of different sorts of objects with different combinations of capabilities, including for example ocean waves, tornados, the planetary system, as well as plants, animals, ecosystems, and many sorts of machines, all of which maintain some sort of coherence over time despite considerable physical changes and despite external perturbances and internal development. These systems exhibit different sorts of stability depending on the variety and intensity of disturbing influences that they can resist.

Where parts cooperate to maintain global relations we can talk of those parts having a *function*, whether they were designed for that function or evolved biologically, or not. Many possible states and processes are consistent with the normal state of such a system, but when a change irreversibly interferes with one or more functions we may speak of "damage". The damaged system has a different set of inherent possibilities. In the normal undamaged configuration both the old and the new sets of possibilities exist. However some sets are "more remote" than others: they cannot be realised without a change in the configuration, whereas the "less remote" possibilities are all able to be actualised without changing the configuration and without damage. A less remote possibility might include a lever changing its orientation. A more remote possibility might include the lever breaking.

The notion of an undamaged configuration needs to be made more precise. In the case of an artefact there will be a set of possible states that will be described as undamaged states of the machine and others that are described as damaged states, the difference being related to the intended use of the machine. For now I wish to disregard such purposive notions and consider only classes of states that are identifiable by

their physical properties.

Instead of talking about damage we can describe a configuration as preserved as long as the components continue to have certain properties and stand in certain relationships. Thus, for a clock, rotation of the hands and the cogwheels to which they are connected preserves a certain sort of configuration, whereas removal of a tooth from a cog does not.

We could clamp two parts of the clock together and define a new configuration as one that includes all the previous relationships and also the clamping relationship. The new configuration will directly support a smaller range of possibilities. Some of the possibilities that were directly supported by the previous configuration are remote relative to the new configuration. However the restriction may also enable new possibilities, such as the unattended clock remaining wound up for a long time.

A configuration then is defined not just by its physical components, but by a particular set of properties and relationships between those components (e.g. those in which all components of the clock retain their shape, their pivot points, the distances between pivot points, etc.).

I shall say that the configuration *directly* supports the collection of possibilities that require no change in the configuration (e.g. no change of shape of the rigid components) and *indirectly* supports larger classes of possibilities, which would require more or less drastic changes in the configuration. We could define a technical notion of "damage" to a configuration as removal of one of its defining relationships. Then achieving one of the more remote possibilities requires damage to the configuration.

What sorts of configurations are worth considering as having "defining" properties is a large question I shall not consider. Living organisms and machines designed by intelligent agents would be obvious examples. Others might be cities, rampaging crowds, galaxies, the solar system, stable complex molecules, tornados, clouds and other naturally occurring complex behaving structures with some degree of integration or coherence. For now I shall merely assume there are some interesting cases without attempting to delimit the whole class.

8 Do we need new forms of representation? De-dicto and de-re modality

So far I have used informal language to talk about possibilities. Can we represent them precisely so that a machine can manipulate them? Modal logic might look like an option. It is concerned with operators that can be applied to sentences to produce new sentences, e.g. "It is possible that P", "It is necessarily the case that P". However, in ordinary language, the adjectives "possible" and "impossible" can be applied directly to objects, events or processes, and not only to propositions. This "de-re" form of words might be construed as an abbreviation for an assertion in which a modal operator is applied to a complete proposition (de-dicto modality).

We should at least consider the alternative hypothesis that there is a more basic notion of possibility than possible truth or falsity of a proposition, namely a property of objects, events or processes (de-re modality). I am making both an ontological claim about the nature of reality, and an epistemological claim about our information processing capabilities. For all the sorts of reasons given above, it seems that de-re modality plays an important role in our perception (e.g. perceiving affordances), thinking and communication, that it is used by animals that cannot construct and manipulate complete propositions, and that it will figure in the internal processes of intelligent robots. However, this remains a *conjecture* open to empirical refutation.

When Gibson claimed that the primary function of biological perceptual systems is to provide information about positive and negative affordances, such as support, obstruction, graspability, etc. I don't think he meant that animal visual systems produce propositions and apply modal operators to them. Is there a different form of representation which captures in a more powerful fashion information which enables the animal to interact fruitfully with the environment?

8.1 Possible world semantics for modal terms

Readers familiar with possible world semantics for modal operators involving notions of degrees of "accessibility" between possible worlds will see obvious links with the notion of more or less remotely supported sets of possibilities. However there is no simple ordering associated with a degree of remoteness of possibilities supported by a configuration.

For example the sets of possibilities that become accessible when one of the levers is broken, when some of the teeth are broken off a cog wheel, when a belt is removed from a pulley need not form a set ordered by inclusion. If we consider different *combinations* of kinds of damage or other change to the configuration, we get a very wide variety of sets of possibilities with at most a partial ordering in degree of remoteness from the set of possibilities directly supported by the original configuration. If there are different sequences of kinds of damage leading to the same state there need not even be a well defined partial ordering. In one sequence getting to state A requires more changes than getting to state B. In another sequence it could be the other way round.

Thus there need not be any well-defined ordering or metric that can be applied to notions of relative degree of remoteness of possibilities, and these ideas will not correspond exactly to a possible world semantics which requires degrees of accessibility to be totally or partially ordered. (Where there is no such requirement the two views of modality may turn out to be equivalent.)

There is a deeper problem about sticking to modal logic and current semantic theories, namely that we have no reason to presuppose that an adequate ontology for science or engineering would have the form of a model for a logical system,

with a well defined set of objects, properties and relationships, so that all configurations can be completely described in the language of predicate calculus, and all changes can be described in terms of successions of logical formulae.

For example the wings of a hovering humming bird or a gliding seagull are very complex flexible structures whose shape is constantly changing in such a way as to change the transduction relationships between various possible forces (gravity, wind pressure, air resistance, etc.) and possible changes of location, orientation and speed. Is there some useful decomposition of the wings into a fixed set of objects whose properties and relationships can be described completely in predicate calculus (augmented if necessary with partial differential equations linking properties, etc.) Or do we need a different more "fluid" sort of representation for the dynamics of such flexible structures. An attempt to represent changing fields of influence diagrammatically can be found in Lundell (1996).

When a child learns how to dress itself can its understanding of the processes of putting on a sweater or tying shoe laces can be expressed in terms some articulation of reality into a form that is expressible in terms of names of individual objects (e.g. well defined parts of the sweater besides the obvious global parts such as sleeves, head opening, etc.), predicates, relations and functions? Or is some totally different form of representation still waiting to be discovered that is required for such tasks (Sloman 1989). Could the neural mechanisms in animal brains teach us about new powerful forms of representation.

9 Possibilities causes and counterfactual conditionals

Physical properties such as electrical resistance, tensile strength, fragility, flexibility, rigidity, all involve relationships between ranges of possibilities, as described above. In some cases the range of possibilities forms a linear continuum (e.g. possible voltages, possible currents) and in those cases the property linking them imposes a constraint that may be expressible in a particularly simple form, such as an equation linking two or more numerical measures. Other sets of possibilities may have more complex structures.

For example, a fragile vase may be struck or crushed in many different ways, and the resulting decomposition into myriad fragments can happen in many different ways. Here the two ranges of possibilities do not form linear sets, and therefore the links between them cannot be expressed as a single simple equation linking two numerical variables. In the case of a digital circuit or a software system there may be ranges of possibilities, both for inputs and outputs, that are not even continuous, for example possible inputs and outputs for a compiler, or possible startup files for an operating system and possible subsequent behaviours.

Despite the diversity, in all the cases there is a relationship between the sets of possibilities which can at least loosely be characterised by saying that the properties of the machine or configuration ensure that IF certain possibilities occur THEN certain others occur. Moreover some of these possibilities may not actually occur: the fragile vase sits in the museum forever without being hit by a sledge hammer. We can still say that the IF-THEN relationship holds for that vase. In that case we are talking about counterfactual conditionals. However, there is no difference in meaning between counterfactual conditions with false antecedent and other conditionals with true antecedent. In both cases the conditionals assert that there is a relationship between elements of certain sets of possibilities. I.e. the truth of a conditional and the existence of a possibility transducer may come to the same thing.

When we say that one thing causes another, as opposed to merely occurring prior to it we are referring implicitly to a set of such conditionals. However, this is a large topic on which huge amounts have been written (e.g. Taylor 1992) and I do not intend to address the problem in this paper, merely to point out the connection. If the previous point about expressions in predicate logic being insufficient to express all the types of possibilities that an intelligent agent needs to perceive and understand is correct, then the notion of a conditional has to be construed as something that does not merely link assertions in familiar logical formalisms. We may have to generalise it to cover new types of representations. (Some feed-forward neural nets can already be seen as an example of something different: namely a large collection of of parallel probabilistic IF-THEN rules. (Sloman 1994))

10 Levels of virtual machines

I previously remarked that a property like electrical resistance may be *implemented* in lower level physical structures, whose properties and relationships would be defined in the language of contemporary physics, or perhaps physics of the future. This is a simple example of a very general phenomenon. There are many properties of complex systems that are best thought of as "implemented" in other properties. This is commonplace in computer science and software engineering, where we often talk about virtual machines, as in the wordprocessor example. Similarly, phenomena typically studied by biologists (including genes) are implemented in mechanisms involving physics and chemistry, and phenomena studied by chemists (e.g. in drug companies) are implemented in mechanisms of quantum mechanics which the chemists may not understand. Social, political, and economic phenomena are also implemented in very complex ways in the physical world, but with many levels of intermediate virtual machines.

The points made previously about physical objects having properties which are essentially causal linkages between ranges of possibilities apply also to objects in virtual machines, including social mechanisms and abstract or virtual machines running on computers. In other words, it is not only the "bottom" level physical constituents of the universe, whatever they may be, that have causal powers: all

sorts of more or less concrete objects also do. In fact a great deal of our normal thinking and planning would be completely impossible but for this fact, for we think about causal connections (e.g. connections between poverty and crime) without having any idea about the detailed physical implementation. Sometimes without knowing the details we can still interact very effectively with relevant bits of the world, like engineers thinking only about current and voltage but not the underlying quantum mechanics. As the vase and gambling machines show, we don't even need to presuppose complete causal determinism.

Admittedly social and economic engineering are very difficult, but we constantly interact with and bring about changes in people and their thought processes, desires, intentions and plans. Similar causal interactions can happen between coexisting abstract states entirely *within* a brain or an office information system.

11 Causal powers of computational states

An important implication of all this is that whereas many people have the intuitive feeling that somehow computational processes are incapable of having the properties required for mental states, events and processes, such as desires, pains, pleasures, emotions, experiences of colour, etc. this may be because they think of computations as if they were simply static structures like collections of symbols on a sheet of paper.

However, when symbolic information structures are implemented in a working system which is actually controlling some complex physical configuration, such as an airliner coming in to land, the limbs of a robot, or the machinery in a chemical plant, then it is a crucial fact about the system that the information processing states and events and the abstract data-structures, all have causal powers both to act on one another and also (through appropriate input and output devices) to change, and be changed by, gross physical structures in the environment. A further development of this line of thought would show how to defend AI theories of mind against charges that no computational system could have the right causal powers to support mental states and processes. But that is a topic for another occasion.

12 Grammars for things and for processes.

Can we move towards a scientific theory of sets of possibilities and how they are related, with sufficient precision to provide a basis for designing robots which perceive and think about possibilities? I suspect we are not yet ready to complete this task, though much work in AI can be seen as addressing it. (Recent examples are Chittaro & Ranon (1996), Lundell (1996), Stahovich et al. (1996) as well as much work on constraint manipulation).

Perhaps the notion of a formal grammar will turn out to be relevant. Grammars are specifications of sets of possibilities: usually sets of legal formulas within a formalism. However various attempts have been made to generalise this notion to accommodate, for example, grammars for images, grammars for 3-D structures, and grammars for behaviours (e.g. dances). I am not aware of any grammatical formalism that is able to cope with some of the kinds of continuous variability mentioned above (e.g. changes of configuration as a child dons a sweater.) Nevertheless it may be that some future development of some existing grammar formalism will suffice, perhaps combined with techniques for constraint propagation.

The main point for now is that the components of the grammar do not need to be propositions or constituents of propositions. So a grammar provides a different view of ranges of possibilities from that provided by a modal logic. Roughly one is object-centred (de-re) and the other fact-centred, or proposition-centred (de-dicto). However it is not clear whether these are trivially equivalent.

13 Conclusion

Many common sense and scientific notions of things are inherently modal (i.e. to do with possibilities and relationships between possibilities), including both explicitly dispositional concepts (e.g. "brittleness", "risk", "irritability"), and many others (e.g. "electrical resistance", "volume", "shape").

I have offered a view of objects (both physical objects and more abstract objects like data-structures or procedures in a virtual machine) as having properties that are inherently connected with sets of possibilities, some of the possibilities being causal inputs to the object and some outputs, and I have suggested that many of the important properties of the objects are concerned with the relationships between possibilities in different sets, i.e. causal links between possibilities. These properties are often *implemented* in lower level properties of different kinds. Moreover, by combining them in larger configurations we can use them to *implement* higher level "emergent" machines, producing many layers of implementation.

In some virtual machines we find causal powers linking events in ways that might later provide detailed models of how human mental processes work. Similar but simpler cases already exist in software systems.

One consequence of this way of thinking is that we don't have to go to quantum mechanics to be faced with issues concerning collections of coexisting possibilities from which reality makes selections.

Acknowledgements

I have benefited from interactions with many people including Natasha Alechina, Pat Hayes, Michael Jampel, Brian Logan, Mark Ryan, Henry Stapp, Bill Robinson, Toby Walsh, Dave Waltz, Ian Wright, Bill Woods, and the anonymous referees. My thinking on these topics is deeply influenced by the work of Gilbert Ryle. The ideas presented

here overlap considerably with those in Bhaskar (1978). Chapter 2 of Sloman (1978) contains an early attempt of my own to address these issues.

References

R Bhaskar (1978) *A Realist Theory of Science* Sussex:The Harvester Press

L. Chittaro & R Ranon (1996) Augmenting the diagnostic power of flow-based approaches to functional reasoning, in *Proc 13th National Conference on AI (AAAI96)* Portland, Oregon, 1010–1015, AAAI Press/MIT Press

J.J. Gibson, (1986) *The Ecological Approach to Visual Perception*, Lawrence Earlbaum Associates, 1986 (originally published in 1979).

J. Holland (1995) em Hidden order: How adaptation builds complexity Reading, Mass: Addison Wesley

M. Lundell (1996) A qualitative model of physical fields, in *Proc 13th National Conference on AI (AAAI96)* Portland, Oregon, 1016–1021, AAAI Press/MIT Press

L. Pryor (1969) Opportunity recognition in complex environments, in *Proc 13th National Conference on AI (AAAI96)* Portland, Oregon, 1147–1152, AAAI Press/MIT Press

G. Ryle (1949) *The Concept of Mind,* Hutchinson.

A. Sloman (1978) *The Computer Revolution in Philosophy: Philosophy Science and Models of Mind* Hassocks: Harvester Press 1978.

A. Sloman (1989) 'On designing a visual system: Towards a Gibsonian computational model of vision' *Journal of Experimental and Theoretical AI* **1**(4), 289–337

A. Sloman (1994), Semantics in an intelligent control system, in *Philosophical Transactions of the Royal Society: Physical Sciences and Engineering*, **349**(1689), 43–58.

T. F. Stahovich, R. Davis & H. Shrobe (1996) Generating multiple new designs from a sketch, in *Proc 13th National Conference on AI (AAAI96)* Portland, Oregon, 1022–1029, AAAI Press/MIT Press

C.N. Taylor (1992) *A Formal Logical Analysis of Causal Relations* DPhil Thesis, Sussex University. Available as Cognitive Science Research Paper No.257

Invited Talks

FROM HERE TO HUMAN-LEVEL AI

John McCarthy
Computer Science Department
Stanford University
Stanford, CA 94305
jmc@cs.stanford.edu
http://www-formal.stanford.edu/jmc/

Abstract

It is not surprising that reaching human-level AI has proved to be difficult and progress has been slow—though there has been definite progress. The slowness and the demand to exploit what has been discovered has led many to mistakenly redefine AI, sometimes in ways that preclude human-level AI—by relegating to humans parts of the task that human-level computer programs should do. In the terminology of this paper, it amounts to settling for a *bounded informatic situation* instead of the more general *common sense informatic situation.*

Overcoming the "brittleness" of present AI systems and reaching human-level AI requires programs that deal with the *common sense informatic situation*—in which the phenomena to be taken into account in achieving a goal are not fixed in advance.

We discuss reaching human-level AI, emphasizing logical AI and especially emphasizing representation problems of information and of reasoning. Ideas for reasoning in the common sense informatic situation include nonmonotonic reasoning, approximate concepts, formalized contexts and introspection.

By the time of the conference there will be a paper with the above title and the URL http://www-formal.stanford.edu/jmc/human.html.

1 What is Human-Level AI?

The first scientific discussion of human level machine intelligence was apparently by Alan Turing in the lecture [Turing, 1947]. The notion was amplified as a goal in [Turing, 1950], but at least the latter paper did not say what would have to be done to achieve the goal.

Allen Newell and Herbert Simon in 1954 were the first people to make a start on programming computers for general intelligence.

Many tasks that humans can do, humans cannot yet make computers do. There are two approaches to human-level AI, but each presents difficulties. It isn't a question of deciding between them, because each should eventually succeed; it is more a race.

1. If we understood enough about how the human intellect works, we could simulate it. However, we don't have have sufficient ability to observe ourselves or others to understand directly how our intellects work. Understanding the human brain well enough to imitate its function therefore requires theoretical and experimental success in psychology and neurophysiology. See [Newell and Simon, 1972] for the beginning of the information processing approach to psychology.

2. To the extent that we understand the problems achieving goals in the world presents to intelligence we can write intelligent programs. That's what this article is about.

What problems does the world present to intelligence? More narrowly, we consider the problems it would present to a human scale robot faced with the problems humans might be inclined to relegate to sufficiently intelligent robots. The physical world of a robot contains middle sized objects about which its sensory apparatus can obtain only partial information quite inadequate to fully determne the effects of its future actions. Its mental world includes its interactions with people and also meta-information about the information it has or can obtain.

Our approach is based on what we call the *common sense informatic situation*. In order to explain the common sense informatic situation, we contrast it with the *bounded informatic situation* that characterizes both formal scientific theories and almost all (maybe all) experimental work in AI done so far.

A formal theory in the physical sciences deals with a *bounded informatic situation*. Scientists decide informally in advance what phenomena to take into account. For example, much celestial mechanics is done within the Newtonian gravitational theory and does not take into account possible additional effects such as outgassing from a comet or electromagnetic forces exerted by the solar wind. If more phenomena are to be considered, a person must make a new theory. Probabilistic and fuzzy uncertainties can still fit into a bounded informatic system; it is only necessary that the set of possibilities (sample space) be bounded.

Most AI formalisms also work only in a bounded informatic situation. What phenomena to take into account is decided by a person before the formal theory is constructed. With such restrictions, much of the reasoning can be monotonic, but such systems cannot reach human level ability. For that, the machine will have to decide for itself what information is relevant. When a *bounded informatic system* is appropriate, the system must construct or choose a limited *context* containing a suitable theory whose predicates and functions connect to the machine's inputs and outputs in an appropriate way. The logical tool for this is *nonmonotonic* reasoning.

2 The Common Sense Informatic Situation

Contention: The key to reaching human-level AI is making systems that operate successfully in the common sense informatic situation.

In general a thinking human is in what we call the *common sense informatic situation* first proposed in[1] [McCarthy,]. It is more general than any *bounded informatic situation*. The known facts are incomplete, and there is no *a priori* limitation on what facts are relevant. It may not even be decided in advance what phenomena are to be taken into account. The consequences of actions cannot be fully determined. The *common sense informatic situation* necessitates the use of *approximate concepts* that cannot be fully defined and the use of *approximate theories* involving them. It also requires *nonmonotonic* reasoning in reaching conclusions.

The common sense informatic situation also includes some knowledge about the system's mental state.

A nice example of the common sense informatic situation is given by the problem of grading solutions to a physics problem that was extensively discussed in the *American Journal of Physics* some years ago. The exam problem is to find the height of a building using a barometer. The intended solution is to measure the air pressure at the top and bottom of the building and multiply the difference by the ratio of the density of mercury to the density of air. However, other answers may be offered. (1) drop the barometer from the top of the building and measure the time before it hits the ground. (2) Measure the height and shadow of the barometer and measure the length of the shadow of the building. (3) Rappel down the building using the barometer as a measuring rod. (4) Lower the barometer on a string till it reaches the ground and measure the string. (5) Offer the barometer to the janitor of the building in exchange for information about the height. (6) Ignore the barometer, count the stories of the building and multiply be ten feet.

Clearly it is not possible to bound in advance the common sense knowledge of the world that may be relevant to grading the problem. Grading some of the solutions requires knowledge of the formalisms of physics and the physical facts about the earth, e.g. the law of falling bodies or the variation of air pressure with altitude. However, in every case, the physics knowledge is embedded in common sense knowledge. Thus before one can use Galileo's law of falling bodies $s = \frac{1}{2}gt^2$, one needs common sense information about buildings, their shapes and their roofs.

Bounded informatic situations are obtained by nonmonotonically inferring that only the phenomena that somehow appear to be relevant are relevant. In the barometer example, the student was expected to infer that the barometer was only to be used in the conventional way for measuring air pressure. For example, a reasoning system might do this by applying circumscription to a predicate *relevant* in a formalism containing also metalinguistic information, e.g. that this was a problem assigned in a physics course. Formalizing relevance in a useful way promises to be difficult.

Common sense facts and common sense reasoning are necessarily imprecise. The imprecision necessitated by the common sense informatic situation applies to computer programs as well as to people.

Some kinds of imprecision can be represented numerically and have been explored with the aid of Bayesian

[1] http://www-formal.stanford.edu/jmc/ailogic.html

networks, fuzzy logic and similar formalisms. This is in addition to the study of approximation in numerical analysis and the physical sciences.

3 The Use of Mathematical Logic

What about mathematical logical languages?

Mathematical logic was devised to formalize precise facts and correct reasoning. Its founders, Leibniz, Boole and Frege, hoped to use it for common sense facts and reasoning, not realizing that the imprecision of concepts used in common sense language was often a necessary feature and not always a bug. The biggest success of mathematical logic have been in formalizing mathematical theories. Since it is necessary to use imprecise facts and imprecise reasoning, the use of mathematical logic for common sense has had limited success. This has caused many people to give up. Gradually extended logical languages and even extended forms of mathematical logic are being invented and developed.

It is necessary to distinguish between mathematical logic and particular mathematical logical languages. Particular logical languages are determined by a particular choice of concepts and the predicate and function symbols to represent them. Failure to make the distinction has often led to error. When a particular logical language has been shown inadequate for some purpose, some people have concluded that logic is inadequate. Different concepts and different predicate and function symbols might still succeed. In the words of the drive-in movie critic of Grapevine, Texas, "I'm surprised I have to explain this stuff."

The pessimists about logic or some particular set of predicates might try to prove a theorem about its inadequacies for expressing common sense.[2]

Since it seems clear that humans don't use logic as a basic internal representation formalism, maybe something else will work better for AI. Researchers have been trying to find this something else since the 1950s but still haven't succeeded in getting anything that is ready to be applied to the common sense informatic situation. Maybe they will eventually succeed.

Mathematical logic has been concerned with how people ought to think rather than how people do think. We who use logic as a basic AI formalism make programs reason logically. However, we have to extend logic and extend the programs that use it in various ways.

One important extension was the development of modal logic starting in the 1920s and using it to treat modalities like knowledge, belief and obligation. Modalities can be treated either by using modal logic or by reifying concepts and sentences within the standard logic. My opinion is that reification in standard logic is more powerful and will work better.

A second extension was the formalization of nonmonotonic reasoning beginning in the late 1970s—with circumscription and default logic and their variants as the major proposals. Nonmonotonic logic has been studied both as pure mathematics and in application to AI problems, most prominently to the formalization of action and causality. Several variants of the major formalisms have been devised.

Success so far has been moderate, and it isn't clear whether greater success can be obtained by changing the the concepts and their representation by predicate and function symbols or by varying the nonmonotonic formalism. [3]

We need to distinguish the actual use of logic from what Allen Newell, [Newell, 1981] and [Newell, 1993], calls the logic level and which was also proposed in **Ascribing Mental Qualities to Machines**[4] [McCarthy, 1979].

4 Approximate Concepts and Approximate Theories

Other kinds of imprecision are more fundamental for intelligence than numerical imprecision. Many phenomena in the world are appropriately described in terms of *approximate concepts*. Although the concepts are imprecise, many statements using them have precise truth values. We offer two examples: the concept of Mount Everest and the concept of the welfare of a chicken. The exact pieces of rock and ice that constitute Mount Everest are unclear. For many rocks, there is no truth of the matter as to whether it is part of Mount Everest. Nevertheless, it is true without qualification that Edmund Hillary and Tenzing Norgay climbed Mount Everest in 1953 and that John McCarthy never set foot on it.

The point of this example is that it is possible and even common to have a solid knowledge structure from which solid conclusions can be inferred based on a

[2] Gödel's theorem is not relevant to this, because the question is not one of decideability or of characterizing truth.

[3] One referee for KR96 foolishly and arrogantly proposed rejecting a paper on the grounds that the inadequacy of circumscription for representing action was known.

[4] http://www-formal.stanford.edu/jmc/ascribing.html

foundation built on the quicksand of approximate concepts without definite extensions.

As for the chicken, it is clear that feeding it helps it and wringing its neck harms it, but it is unclear what its welfare consists of over the course of the decade from the time of its hatching. Is it better off leading a life of poultry luxury and eventually being slaughtered or would it be better off escaping the chicken yard and taking its chances on foxes? There is no truth of the matter to be determined by careful investigation of chickens. **When a concept is inherently approximate, it is a waste of time to try to give it a precise definition.** Indeed different efforts to define such a concept precisely will lead to different results.

Most human common sense knowledge involves approximate concepts, and reaching human-level AI requires a satisfactory way of representing information involving approximate concepts.

5 Nonmonotonic Reasoning

Common sense reasoning is also imprecise in that it draws conclusions that might not be made if there were more information. Thus common sense reasoning is *nonmonotonic*. I will not go into the details of any of the proposals for handling nonmonotonic reasoning, supposing them to be substantially familiar to most of the readers of these proceedings. In particular, getting from the common sense informatic situation to a bounded informatic situation needs nonmonotonic reasoning.

Unfortunately, there is not time to discuss further what developments in nonmonotonic reasoning must lie between the present situation and human-level AI.

6 Elaboration Tolerance

Human abilities in the common sense informatic situation also include what may be called *elaboration tolerance*—the ability to elaborate a statement of some facts without having to start all over. Thus when we begin to think about a problem, e.g. determining the height of a building, we form a bounded context and try to solve the problem within it. However, at any time more facts can be added, e.g. about the precision with which the time for the barometer to fall can be estimated using a stop watch and also the possibilities of acquiring a stop watch.

A draft on **elaboration tolerance**[5] [McCarthy and Amir, 1996] gives more detail. It discusses about 25 elaborations of the Missionaries and Cannibals problem.

What I have so far said so far about approximate concepts, nonmonotonic reasoning and elaboration tolerance is independent of whether mathematical logic, human language or some other formalism is used.

In my opinion, the best AI results so far have been obtained using and extending mathematical logic.

7 Formalization of Context

A third extension of mathematical logic involves **formalizing the notion of context**[6] [McCarthy, 1993]. Notice that when logical theories are used in human communication and study, the theory is used in a context which people can discuss from the outside. If computers are to have this facility and are to work within logic, then the "outer" logical language needs names for contexts and sentences giving their relations and a way of entering a context. Clearly human-level AI requires reasoning about context.

Human-level AI also requires the ability to *transcend* the outermost context the system has used so far. Besides in [McCarthy, 1993], this is also discussed in **Making Robots Conscious of their Mental States**[7] [McCarthy, 1996].

Further work includes [Buvač, 1996] and [Buvač et al., 1995].

8 Reasoning about Events—Especially Actions

Reasoning about actions has been a major AI activity, but this paper will not discuss my or other people's current approaches, concentrating instead on the long range problem of reaching human level capability. We regard actions as particular kinds of events and therefore propose subsuming reasoning about actions under the heading of reasoning about events.

Most reasoning about events has concerned determining the effects of an explicitly given sequence of actions by a single actor. Within this framework various problems have been studied.

- The frame problem concerns not having to state what does not change when an event occurs.
- The qualification problem concerns not having to state all the preconditions of an action or other

[5] http://www-formal.stanford.edu/jmc/elaboration.html

[6] http://www-formal.stanford.edu/jmc/context.html

[7] http://www-formal.stanford.edu/jmc/consciousness.html

event. The point is both to limit the set of preconditions and also to jump to the conclusion that unstated others will be fulfilled unless there is evidence to the contrary. For example, wearing clothes is a precondition for airline travel, but the travel agent will not tell his customer to be sure and wear clothes.

- The ramification problem concerns how to treat side-effects of events other than the principal effect mentioned in the event description.

Each of these involves elaboration tolerance, e.g. adding descriptions of the effects of additional events without having to change the descriptions of the events already described. When I wrote about **applications of circumscription to formalizing common sense**[8] [McCarthy, 1986], I hoped that a *simple abnormality theory* would suffice for all of them. That didn't work out when I tried it, but I still think a common nonmonotonic reasoning mechanism will work. Tom Costello's draft *"The Expressive Power of Circumscripttion"*[9] argues that simple abnormality theories have the same expressive power as more elaborate nonmonotonic formalisms that have been proposed.

Human level intelligence requires reasoning about strategies of action, i.e. action programs. It also requires considering multiple actors and also concurrent events and continuous events. Clearly we have a long way to go.

Some of these points are discussed in a draft on narrative[10] [McCarthy, 1995].

9 Introspection

People have a limited ability to observe their own mental processes. For many intellectual tasks introspection is irrelevant. However, it is at least relevant for evaluating how one is using one's own thinking time. Human-level AI will require introspective ability.

That robots also need introspection[11] is argued and how to do it is discussed in [McCarthy, 1996].

10 Heuristics

The largest qualitative gap between human performance and computer performance is in the area of heuristics, even though the gap is disguised in many applications by the millions-fold speed advantage of computers. The general purpose theorem proving programs run very slowly, and the special purpose programs are very specialized in their heuristics.

I think the problem lies in our present inability to give programs domain and problem dependent heuristic advice. In my Advice Taker paper[12] [McCarthy, 1958] I advertised that the Advice Taker would express its heuristics declaratively. Maybe that will work, but neither I nor anyone else has been able to get a start on the problem in the ensuing almost 40 years. Another possibility is to express the advice in a *procedure modification language*, i.e. to extend elaboration tolerance to programs. Of course, every kind of modularity, e.g. object orientation, gives some elaboration tolerance, but these devices haven't been good enough.

Ideally, a general purpose reasoning system would be able to accept advice permitting it to run at a fixed ratio speed of speeds to a special purpose program, e.g. at 1/20 th the speed.

11 Summary

Conclusion: Between us and human-level intelligence lie many problems. They can be summarized as that of succeeding in the *common sense informatic situation.*

The problems include:

common sense knowledge of the world Many important aspects of what this knowledge is in and how it can be represented are still unsolved questions. This is particularly true of knowledge of the effects of actions and other events.

epistemologically adequate languages These are languages for expressing **what a person or robot can actually learn about the world**[13] [McCarthy and Hayes, 1969].

elaboration tolerance What a person knows can be elaborated without starting all over.

nonmonotonic reasoning Perhaps new systems are needed.

contexts as objects This subject is just beginning. See the references of section 7.

introspection AI systems will need to examine their own internal states.

[8] http://www-formal.stanford.edu/jmc/applications.html
[9] http://www-formal.stanford.edu/tjc/expressive.html
[10] http://www-formal.stanford.edu/jmc/narrative.html
[11] http://www-formal.stanford.edu/jmc/consciousness.html
[12] http://www-formal.stanford.edu/jmc/mcc59.html
[13] http://www-formal.stanford.edu/jmc/mcchay69.html

action The present puzzles of formalizing action should admit a uniform solution.

I doubt that a human-level intelligent program will have structures corresponding to all these entities and to the others that might have been listed. A generally intelligent logical program probably needs only its monotonic and nonmonotonic reasoning mechanisms plus mechanisms for entering and leaving contexts. The rest are handled by particular functions and predicates.

12 Remarks and Acknowledgements

1. To what extent will all these problems have to be faced explicitly by people working with neural nets and connectionist systems? The systems I know about are too primitive for the problems even to arise. However, more ambitious systems will inhabit the common sense informatic situation. They will have to be elaboration tolerant and will require some kind of mental model of the consequences of actions.

2. This work was partly supported by ARPA (ONR) grant N00014-94-1-0775.

3. I got useful suggestions from Eyal Amir, Saša Buvač and Tom Costello.

4. Some additional relevant papers are in my book [McCarthy, 1990] and on my Web site[14].

5. My understanding that I should prepare a printable version of this invited talk came rather late. I expect that both the spoken version and the 1996 November Web version will have better explanations of the important concepts.

References

[Buvač, 1996] Buvač, S. (1996). Quantificational logic of context. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence.*

[Buvač et al., 1995] Buvač, S., Buvač, V., and Mason, I. A. (1995). Metamathematics of contexts. *Fundamenta Informaticae*, 23(3).

[McCarthy,] McCarthy, J. Artificial intelligence, logic and formalizing common sense. In Thomason, R., editor, *Philosophical Logic and Artificial Intelligence.* Klüver Academic.

[14] http://www-formal.stanford.edu/jmc/

[McCarthy, 1958] McCarthy, J. (1958). Programs with common sense. In *Mechanisation of Thought Processes, Proceedings of the Symposium of the National Physics Laboratory*, pages 77–84, London, U.K. Her Majesty's Stationary Office. Reprinted in McC90.

[McCarthy, 1979] McCarthy, J. (1979). Ascribing mental qualities to machines. In Ringle, M., editor, *Philosophical Perspectives in Artificial Intelligence.* Harvester Press. Reprinted in [?].

[McCarthy, 1986] McCarthy, J. (1986). Applications of circumscription to formalizing common sense knowledge. *Artificial Intelligence*, 28:89–116. Reprinted in [McCarthy, 1990].

[McCarthy, 1990] McCarthy, J. (1990). *Formalization of common sense, papers by John McCarthy edited by V. Lifschitz.* Ablex.

[McCarthy, 1993] McCarthy, J. (1993). Notes on formalizing context. In *IJCAI-93.* Available on http://www-formal.stanford.edu/jmc/.

[McCarthy, 1995] McCarthy, J. (1995). Situation calculus with concurrent events and narrative. available at URL: http://www-formal.stanford.edu/jmc/ Contents subject to change. Reference will remain.

[McCarthy, 1996] McCarthy, J. (1996). Making robots conscious of their mental states. In Muggleton, S., editor, *Machine Intelligence 15.* Oxford University Press. to appear, available on http://www-formal.stanford.edu/jmc/.

[McCarthy and Amir, 1996] McCarthy, J. and Amir, E. (1996). Missionaries and cannibals: Making it elaboration tolerant. *McCarthy Web page http://www-formal.stanford.edu/jmc/.* Available as http://www-formal.stanford.edu/jmc/elaboration.html.

[McCarthy and Hayes, 1969] McCarthy, J. and Hayes, P. J. (1969). Some philosophical problems from the standpoint of artificial intelligence. In Meltzer, B. and Michie, D., editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press.

[Newell, 1981] Newell, A. (1981). The knowledge level. *AI Magazine*, 2(2):1–20. Originally delivered as the Presidential Address, American Association for Artificial Intelligence, AAAI80, Stanford, CA, August 1980.

[Newell, 1993] Newell, A. (1993). Reflections on the knowledge level. *Artificial Intelligence*, 59(1-2):31–38.

[Newell and Simon, 1972] Newell, A. and Simon, H. A. (1972). *Human Problem Solving.* Prentice–Hall, Englewood Cliffs, NJ.

[Turing, 1950] Turing, A. (1950). Computing machinery and intelligence. *Mind.*

[Turing, 1947] Turing, A. M. (1947). Lecture to the london mathematical society. In *The Collected Works of A. M. Turing*, volume Mechanical Intelligence. North-Holland. This was apparently the first public introduction of AI, typescript in the King's College archive, the book is 1992.

Complexity and Expressive Power of KR Formalisms

(Invited Talk Abstract)

Georg Gottlob
Institut für Informationssysteme, TU Wien
Paniglgasse 16, A-1040 Wien, Austria
gottlob@dbai.tuwien.ac.at

The complexity of a large number of knowledge representation formalisms was studied during the last five years. Among others, the following logics or techniques were analyzed: default logic [25, 44, 39, 24, 27], autoepistemic logic [25, 14, 23], nonmonotonic modal logics [25, 42, 23], circumscription [8, 17], logic programming (including disjunctive LP) [45, 32, 1, 12, 16, 11, 33], reasoning about temporal relations [36], abduction [4, 15, 20], planning [5, 35], theory revision, and counterfactual reasoning [34, 13, 18]. For a survey, see [9]. Complexity results for both the propositional case and the function-free first order case were derived and recursion-theoretic characterizations for the general case were obtained [10, 2, 30, 31, 40, 41]. In addition, relevant results on approximate reasoning and on the intertranslation between various formalisms were shown.

In this talk, a brief overview of (a selection of) these results is given, and a few key results are explained in detail. It is argued that the complexity-analysis, in addition to expressing a quantitative measure of the worst-case behaviour, leeds to a deeply qualitative understanding of the algorithmic nature of KR reasoning problems. Moreover, by applying methods of descriptive complexity theory (a subfield of finite model theory), we are able to determine the precise expressive power of several KR logics. Latest results are discussed, and directions for future research are given.

There are several reasons, why one should be interested in studying the (worst case) complexity of KR formalisms.

The worst-case complexity is a very good indicator of how many sources of structural complexity are inherent in a problem. For example, if a problem is NP-complete, then it contains basically one source of intractability, related to a choice to be made among exponentially many candidates. If a problem is Σ_2^P-complete, i.e., one level higher than NP in the Polynomial Hierarchy, then there are usually two intermingled sources of complexity. For instance, the problem of checking whether a formula φ is in an extension of a propositional default therory is Σ_2^P complete. The two sources of complexity are (1) the choice of a suitable set G of generating defaults and (dependent on this choice) (2) the proof that G is valid and that the corresponding extension entails φ.

Moreover, once the sources of complexity are identified, one can develop smart algorithms that take these sources into account. In addition, it becomes easier to discover tractable (i.e., polynomial) subcases by considering syntactic restrictions that eliminate all sources of intractability.

Finally, a precise complexity classification gives us valuable information about the algorithmic similarity and intertranslatability of different problems. For instance, both cautious reasoning in autoepistemic logic and cautious reasoning with disjunctive logic programs are Π_2^P-complete. Therefore, there are simple (i.e., polynomial) translations between these two reasoning tasks. In particular, this means that if one has implemented a reasoning engine (theorem prover) for one of these formalisms, this system can easily be adapted to become a reasoning engine for the other formalism. In most cases, the translations between two decision problems that are complete for the same complexity class can be easily be deduced from the respective completeness proofs. At least, the underlying intuitions in these proofs may help to find a suitable translation scheme. On the other hand, if it is known that two problems are complete for different complexity classes in the polynomial hierarchy, the existence of a polynomial translation from the harder to the easier problem is unlikely. For example, it is known that cautious reasoning with non-disjunctive logic programs under the stable model semantics is co-NP-complete [32]. Therefore, unless the polynomial hierarchy collapses, a polynomial translation from disjunctive logic programming (under the stable model semantics) to nondisjunctive logic programming (under the stable model semantics) cannot exist.

In summary, the complexity analysis of a problem

gives us much more than merely a quantitative statement about its tractability or intractability in the worst case. Rather, locating a problem at the right level in the polynomial hierarchy gives us a deep *qualitative* knowledge about this problem.

A finer grained level of comparison between propositional formalisms is obtained through the study of logical translations. For example, it can be seen that even though there exist polynomial translations from default logic to autoepistemic logic [26, 43], there is no *modular* translation [26]. Moreover, by interesting results in [22], Boolean logic is weaker in expressive power than circumscription, which is weaker than Default logic, which in turn, is weaker than preferential default logic (see also [38]).

Computational complexity can also be fruitfully applied to nonpropositional formalisms, i.e., to first and higher order logical systems for knowledge representation. In this case, powerful methods of *finite model theory* can be used to classify the expressive power of such systems over finite structures, i.e., finite bases of factual data.

Since Fagin's famous discovery that the (generic) properties of finite structures expressible in existential second order logic are precisely those computable in nondeterministic polynomial time [21], finite model theory has become one of the most important tools for the analysis of logic-based formalisms in computer science. In particular, the most useful branch of finite model theory in this context is *descriptive complexity theory*. Given a language $\mathcal{L}$ that can express properties over finite structures (e.g. relational databases), describe the set of *all properties expressible in* $\mathcal{L}$ by use of complexity classes. By methods of descriptive complexity theory, we can show that disjunctive logic programming and default logic are more expressive than classical reasoning or normal logic programming [20, 6]. In other terms, there are problems that can be solved with the former but not with the latter formalisms.

Interestingly, the difference in expressive power vanishes when infinite structures are considered. For example, it was recently shown that in presence of function symbols, well-founded logic programs, normal logic programs and disjunctive logic programs all have the same expressive power over infinite Herbrand structures [41, 40, 19]

Other interesting results discussed in the talk are on *knowledge compilation* and *Horn approximation* [7, 28, 29, 38], and on NP approximation algorithms (see Section 13 in [37]).

References

[1] R. Ben-Eliyahu and R. Dechter. Propositional Semantics for Disjunctive Logic Programs. In *Proc. JICSLP-92*, 1992.

[2] H. Blair, W. Marek, A. Nerode, and J. Remmel, editors. *Informal Proceedings of the Workshop on Structural Complexity and Recursion-Theoretic Methods in Logic Programming*, Washington DC, November 1992. Cornell University, Mathematical Sciences Institute.

[3] P. A. Bonatti and T. Eiter. Querying Disjunctive Databases Through Nonmonotonic Logics. To appear in *Theor. Computer Science.*

[4] T. Bylander, D. Allemang, M. Tanner, and J.R. Josephson. Some Results Concerning The Computational Conmplexity of Abduction. *Proc. KR-89*, pp.44-54, 1989.

[5] T. Bylander. Complexity Results for Planning. *Proc. IJCAI-91*, Sydney, Australia, pp.274–279, 1991,

[6] M. Cadoli, T. Eiter, and G. Gottlob. Using Default Logic as a Query Language. In *Proc. KR'94*, 1994. Full paper to appear in *IEEE Transactions on Knowledge and Data Engineering.*

[7] M. Cadoli, F. Donini, M. Schaerf. On Compact Representations of Propositional Circumscription. *Proc. STACS-95*, pp. 205–216,1995.

[8] M. Cadoli and M. Lenzerini. The Complexity of Closed World Reasoning and Circumscription. *Journal of Computer and System Sciences*, 1994.

[9] M. Cadoli and M. Schaerf. A Survey of Complexity Results for Non-monotonic Logics. *Journal of Logic Programming*, 17:127–160, 1993.

[10] J. Chomicki and V. S. Subrahmanian. Generalized Closed World Assumption is $\prod_2^0$-Complete. *Information Processing Letters* 34, 1990.

[11] J. Dix, G. Gottlob, and W. Marek. Causal Models for Disjunctive Logic Programs. In *Proceedings ICLP-94*, 1994.

[12] J. Dix and M. Müller. Abstract Properties and Computational Complexity of Semantics for Disjunctive Logic Programs. In [2].

[13] T. Eiter and G. Gottlob. On the Complexity of Propositional Knowledge Base Revision, Updates, and Counterfactuals. *Artificial Intelligence*, 57(2–3):227–270, 1992.

[14] T. Eiter and G. Gottlob. Reasoning with Parsimonious and Moderately Grounded Expansions. *Fundamenta Informaticae*, 17(1,2):31–53, 1992.

[15] T. Eiter and G. Gottlob. The Complexity of Logic-Based Abduction. *Journal of the ACM* 42(1):3-42, 1995.

[16] T. Eiter and G. Gottlob. On the Computational Cost of Disjunctive Logic Programming: Propositional Case. *Annals of Mathematics and Artificial Intelligence*, vol.15, pp.289–323, 1995.

[17] T. Eiter and G. Gottlob. Propositional Circumscription and Extended Closed World Reasoning are Π_2^P-complete. *Theoretical Computer Science*, 114(2):231–245, 1993. Addendum 118:315.

[18] T. Eiter and G. Gottlob. The Complexity of Nested Counterfactuals and Iterated Knowledge Base Revisions. In R. Bajcsy, editor, *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 526–531. Morgan Kaufman, 1993. Full version to appear in *Journal of Computer ans Syst. Sci.*, 1997.

[19] T. Eiter and G. Gottlob. Expressiveness of Stable Model Semantics for Disjunctive Logic Programming with Function Symbols. *Submitted for publication*, 1996.

[20] T. Eiter, G. Gottlob, and H. Mannila. Adding Disjunction to Datalog. In *Proceedings of the Thirteenth ACM SIGACT SIGMOD-SIGART Symposium on Principles of Database Systems (PODS-94)*, pages 267–278, May 1994.

[21] R. Fagin. Generalized First-Order Spectra and Polynomial-Time Recognizable Sets. In R. M. Karp, editor, *Complexity of Computation*, pages 43–74. AMS, 1974.

[22] G. Gogic, H. Kautz, Ch. Papadimitriou, B. Selman. The Comparative Linguistics of Knowledge Representation. *Proc. IJCAI-95.*

[23] G. Gottlob. NP Trees and Carnap's Modal Logic. *Journal of the ACM* 42(2):421-457, 1995.

[24] G. Gottlob. The Complexity of Propositional Default Reasoning Under the Stationary Fixed Point Semantics. *Inform. and Comput.* 121(1):81–92, 1995.

[25] G. Gottlob. Complexity Results for Nonmonotonic Logics. *Journal of Logic and Computation*, 2(3):397–425, June 1992.

[26] G. Gottlob. Translating Default Logic into Standard Autoepistemic Logic. *JACM*, 42(4):711–740, 1995.

[27] G. Gottlob and Zhang Mingyi. Cumulative Default Logic: Finite Characterization, Algorithms, and Complexity. *Artif Intell.*, 69:329–346, 1994.

[28] D. Kavvadias, Ch. Papadimitriou, M. Sideri. On Horn Envelopes and Hypergraph Transversals *Proc. 1993 ISAACS*, Springer Verlag, 1993.

[29] H. Kautz and B. Selman Knowledge Compilation and Theory Approximation *JACM*, to appear.

[30] W. Marek, A. Nerode, and J. Remmel. A Theory of Nonmonotonic Rule Systems II. *Annals of Mathematics and Artificial Intelligence*, 5:229–264, 1992.

[31] W. Marek, A. Nerode, and J. Remmel. How Complicated is the Set of Stable Models of a Recursive Logic Program? *Annals of Pure and Applied Logic*, 56:119, 1992.

[32] W. Marek and M. Truszczyński. Autoepistemic Logic. *Journal of the ACM*, 38(3):588–619, 1991.

[33] J. Minker. Logic Programming and Databases: A 20 Year Retrospective. In *Proc. Intl. Workshop on Logic in Databases*, San Miniato, Italy, June/July 1996, pp. 5–52, preprint; final version forthcoming in Springer LNCS, 1996.

[34] B. Nebel. Belief Revision and Default Reasoning. In *Proceedings KR-91*, pp.417–428, 1991.

[35] B. Nebel and Ch. Bäckström. On the Computational Complexity of Temporal Projection, Planning, and Plan Validation. In *Artif. Intelligence*, vol. 66, pp.125–160, 1994.

[36] B. Nebel and H.-J. Bürckert. Reasoning about Temporal Relations: A Maximal Tractable Subclass of Allen's Interval Algebra. In *JACM*, 42(1), pp.43–66, 1995.

[37] Ch. Papadimitriou. *Computational Complexity* Addison Wesley, 1994.

[38] Ch. Papadimitriou. The Complexity of Knowledge Representation *Proc. 11th, Annual IEEE Conf. on Computational Complexity*, pp. 244–248, 1996.

[39] C. Papadimitriou and M. Sideri. On Finding Extensions of Default Theories. In *Proc.ICDT-92*, October 1992.

[40] J. Schlipf. A Survey of Complexity and Undecidability Results in Logic Programming. In Blair et al. [2], pages 93–102.

[41] J. Schlipf. The expressive powers of the logic programming semantics. *Journal of Computer and Syst. Sci.*, 51(1):64–86,1995.

[42] G.F. Schwarz, and M. Truszczyński. Nonmonotonic reasoning is sometimes easier. Proc. !l!!of the Kurt Gödel Symposium, pp. 313 – 324, *Lecture Notes in Computer Science*, Springer-Verlag.

[43] G. Schwarz. On embedding Default Logic into Moore's Autoepistemic Logic. *Artificial Intelligence*, vol. 80, pp. 349–359, 1996.

[44] J. Stillman. The Complexity of Propositional Default Logic. In *Proc. AAAI-92*, pages 794–799, 1992.

[45] A. Van Gelder, K.A. Ross, and J.S. Schlipf. Unfounded sets and well-founded semantics for general logic programs. *Journal of the ACM*, 38:620 – 650, 1991.

Panels

Ontologies: What Are They, and Where's The Research?

Panel Abstract and Position Paper

Richard Fikes
Knowledge Systems Laboratory
Department of Computer Science
Stanford University
Stanford, CA 94305

1 INTRODUCTION

In recent years, the AI community has been evolving a notion of ontologies as artifacts that play significant roles in knowledge representation and reasoning (KR&R). The objective of this panel is to contribute to that evolutionary process and to promote a greater degree of consideration of ontology-related research in the KR&R community.

The panel will focus on the following two questions:

- What is a definition of "ontology" that is both precise and useful for AI?
- What are the primary research issues involved in enabling the effective creation and use of ontologies?

We adopt the premise that it would be worthwhile for the KR&R community to converge on a formal definition of what it means by an "ontology". We are particularly interested in precisely relating ontologies to symbol-level artifacts and to the roles they play in KR&R so that ontologies can be distinguished from arbitrary knowledge bases, the formal properties of ontologies can be studied, reasoning methods specifically for ontologies can be developed, effective tools for building and maintaining ontologies can be developed, etc.

The following comments represent some of my personal views on these issues.

2 ONTOLOGIES: WHAT ARE THEY?

Ontologies are generally considered to provide definitions for the vocabulary used to represent knowledge. However, definitions in standard logic languages only enable occurrences of the symbol being defined to be replaced by expressions containing other symbols. Sets of definitions necessarily "bottom out" in expressions containing undefined (primitive) symbols. The undefined symbols are typically given "meaning" by requiring that any interpretation of the symbols satisfy a given set of sentences (axioms) in the conceptualization being represented. Such sentences are not definitions in the classical sense and often constrain more than one of the primitive symbols so that there is no compelling rationale for exclusively associating the sentences with any one symbol. Such sentences must be included in an ontology if it is to "define" the knowledge representation vocabulary. Indeed, one might argue that these sentences that restrict the possible interpretations of undefined symbols are the essential portion of an ontology. Ontologies must therefore contain both sentences and definitions.

What kinds of sentences can be included in an ontology? There does not seem to be any precise way of differentiating between sentences that are "definitional" and sentences that express "contingent facts". Nor does there seem to be any rationale for prohibiting the inclusion of any sentence in an ontology that restricts the interpretations of the symbols occurring in it since any such sentence can be considered to contribute to the specification of the "meaning" of those symbols. It therefore seems that *any* sentence that is not a tautology and that is satisfied in the conceptualization being represented by the intended interpretation is suitable for inclusion in an ontology.

If an ontology can be any set of sentences and definitions, then what is the significance of the "ontology" notion? I suggest that the significance of ontologies is not in what they are, but in the role that they play in representing knowledge. In particular, I suggest that we consider an ontology to be an integral part of a declarative knowledge representation language. That is, consider a declarative knowledge representation language to provide a syntax, a set of inference rules, a vocabulary of non-logical symbols, *and an ontology* that restricts the acceptable interpretations of the symbols in the vocabulary.

If a representation language is assumed to include an ontology, then any knowledge base expressed in a given representation language will implicitly include the axioms and definitions from the language's ontology. Also, when agents agree on the representation language to be used in an upcoming interaction, they will have agreed on an ontology and therefore on a set of mutually shared assumptions about the vocabulary to be used in the interaction.

"Domain-specific" representation languages typically include a domain-specific vocabulary and therefore can naturally be considered to include an ontology. Note that even "domain independent" representation languages often include a vocabulary of relations and functions and have implicit ontologies associated with them. For example, frame languages typically includes vocabulary such as "subclass of", "instance of", "slot value type", and "slot cardinality"; and implementations of those languages assume a set of axioms (e.g., stating that "subclass" is transitive) regarding that vocabulary. Indeed, the entire frame language in our Ontolingua system [Farquhar, et al 96] is defined as an ontology added to KIF [Genesereth & Fikes 92], and KIF itself has been redefined as a core language augmented with ontologies for sets, numbers, lists, etc.

3 ONTOLOGIES: WHERE'S THE RESEARCH?

Ontology creation and use inherits all the research issues of knowledge base creation and use. However, the role that ontologies play in knowledge representation provides important distinguishing features for those issues. For example, since ontologies are not intended to change during the use of the representation language they are a part of, they are particularly suited for "compiling" into reasoning methods and acquisition tools.

A primary difference between ontologies and arbitrary knowledge bases is that there is a much greater emphasis in ontologies on usability in multiple tasks and multiple situations since the ontology is intended to be part of a representation language. Thus, for example, the emphasis is on describing classes of objects rather than individuals and on representing general properties of relations and functions rather than on describing specific situations.

Much has been said about the need to develop reusable encoded knowledge in order to enable the development of large scale intelligent systems. (See, for example, (Patil et al 92).) Ontologies are intended for multiple uses and are therefore an appropriate focus for research on techniques for knowledge reuse.

Adopting reusability as a primary goal for ontologies has a significant impact on the tools and methodologies that are needed for ontology creation and use. For example, developers need to make their ontologies accessible and understandable to a community of use. So, new techniques are needed for translating ontologies between representation formalisms and for describing the competency of an ontology. Also, when knowledge is encoded specifically for use throughout a community, one would expect there to typically be involvement by that community in the encoding process. So, tools that support collaborative development and evolution of ontologies would appear to be important in achieving desired levels of reusability.

There is an apparently inherent tension in reuse of represented knowledge (as there is in the reuse of any software) between level of detail and breadth of applicability. The knowledge that may be appropriate to assume as part of the representation language will vary dramatically from application to application. Thus, even though it is very appealing to have large general-purpose ontologies to include in our representation languages, it seems critically important for ontologies to be available in small composable modules so that the knowledge that is appropriate to assume for a given use can be assembled. A key research challenge, then, is to address the tension between depth and breadth in knowledge reusability. A promising approach to meeting that challenge is to develop both broadly applicable ontologies containing "common sense" knowledge that can be included in general-purpose representation languages and a capability for augmenting that knowledge by assembling composable modules retrieved from online libraries.

Acknowledgments

This work was supported by the Defense Advanced Research Projects Agency under contract N6600196C8622 (ARPA Order D665). I would like to thank Sasa Buvac, Adam Farquhar, and Bill Mark for help with the development of the ideas in this paper.

References

A. Farquhar, R. Fikes, & J. Rice; *The Ontolingua Server: a Tool for Collaborative Ontology Construction*; Proceedings of the Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop; Banff, Canada; November 9-14, 1996.

M. Genesereth and R. Fikes; *Knowledge Interchange Format, Version 3.0 Reference Manual*; Technical Report Logic-92-1, Computer Science Department, Stanford University, Stanford, CA, 1992.

R. Patil, R. Fikes, P. Patel-Schneider, D. Mckay, T. Finin, T. Gruber, R. Neches; *The DARPA Knowledge Sharing Effort: a Progress Report*; in Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning; Cambridge, Massachusetts; October 25-29, 1992. Also KSL Technical Report KSL 93-23.

Ontologies as the Representation (and Re-Representation) of Agreement

Position Paper for the Panel: "Ontologies: What Are They, and Where's the Research?"

William S. Mark
Architecture Laboratory
National Semiconductor Corporation
Santa Clara, CA 94052

1 INTRODUCTION

As a veteran of a number of ontology workshops, I can attest that ontologies are a slippery subject. There is disagreement about their definition, role in knowledge based systems, and their technology (i.e., how they should be built and used). I will use my time on the panel, and this brief note, to argue a few points:

We should bound ontologies by creating them to support a specific set of inter-agent interactions.

Ontologies are needed because independent agents (human, software, or both) share knowledge. As Tom Gruber and Nicola Guarino have argued, ontologies are representations of the agents' agreements about the set of concepts that underlie the information to be shared. I will argue that it is useful to think of ontologies in terms of the inter-agent interactions they need to support. Ontologies must specify enough information about shared concepts to enable the agents to behave appropriately when they receive the designated interactions. They need to be broad enough to include all of the concepts in the set of interactions and deep enough to clearly distinguish the behaviors that should occur when the interactions are received. I think that we may be able to create a design methodology for ontologies built around specifying them in terms of the interactions they must support.

We should design ontologies to be useful to agent developers.

I will argue that, by analogy to standards, ontologies become useful when interfaces are built to handle the interactions they support. Following the analogy, ontologies must be expressed in a form that allows agent developers (in particular, agent interface developers) to understand how to make their agents behave when they receive an interaction supported by the ontology.

We need research into how to adequately support the evolution of ontologies after their initial implementation.

Any implemented ontology is just a starting point: it is impossible to see all of the ramifications of the initial set of agreements. As agents begin to interact, there will inevitably be some shared concepts that need to be further elaborated in order to really express how they are being used. As technology changes, new meanings for shared concepts will develop as existing concepts are used in new ways. Supporting this process requires research progress in at least two areas:

- **representation for ontologies as a set of agreements;**
- **methods for propagating the ramifications of those agreements.**

The rest of this note motivates these points in the framework of the panel questions.

2 ONTOLOGIES: WHAT ARE THEY?

The panel will make an effort to define "ontology", and this will be valuable — more from what we will learn from the discussion than from the precise definition that results. One theme that I think will emerge is that an ontology can be thought of as a set of agreements about a set of concepts.

I would like to emphasize the *process* nature of ontology creation. In particular I compare ontology creation to standards creation. Standards arise because of the need of independently developed elements (hardware or software) to interact with each other. A decision must

be made on exactly what to standardize, i.e., what must be agreed upon and what can remain unshared. This decision is the result of an often lengthy process of *design*, with both technical and political design constraints. (De facto standards are simply preexisting designs that become adopted before other participants enter the discussion). Usually the design is intended to give the standard a "lowest common denominator" character: there is an attempt to keep complexity, or at least detail, in the individual components rather than in the standard. Almost always the standard is subdivided into levels or is otherwise segmented in order to simplify the design problem or reduce the number of stake holders. The standard is then implemented when the participants create interfaces to their elements that cause them to behave in a specified way when a specified interaction occurs.

So:

- Standards are designs for how components should behave when certain interactions occur.
- A key issue is bounding the set of interactions that are covered by the standard, with a goal of keeping the standard at the lowest common denominator.
- Standards are almost always divided into levels or other segments.
- Standards have real impact when they are adhered to by real interfaces.

I think that there are lessons to be learned — positive and negative — from the standards process. On the positive side, I think that the ontologies community should consider how bounding the set of included interactions, use of levels or other segmentation, and interface implementation map into the world of knowledge-based systems. It might be worth looking at ontology creation as a design process whose objective is to focus on what *must* be shared to support a set of interactions. Levels and other forms of segmentation can help to achieve this objective by isolating subsets of the interactions that involve different sets of components. Building ontologies to support interface design provides solid criteria for determining whether specific ontologies are broad and deep enough, appropriately expressed, etc.

Of course, standards have negative aspects as well. First, standards take a long time to create. Second, they often end up acting as a barrier to necessary change. The first problem arises because it is hard to get agreement and even harder to specify exactly what has been agreed upon. The second problem arises because it is impossible to see all of the ramifications of the initial set of agreements. As components begin to interact, there will inevitably be some shared concepts that need to be further elaborated in order to really express how they are being used. As technology changes, new meanings for shared concepts will develop. For example, use of the Internet for transmitting video and speech data has changed the concept of "packet". Packets containing multimedia data must be treated differently by the interacting components of the Internet system (viz., they must be delivered within certain time boundaries and in certain sequences). The transport level of the Internet standard must be changed to cope with this expanded meaning.

I believe that we have the potential to greatly ameliorate the negative aspects of standards. In terms of the first problem, while we can't help with getting agreement, we should be able to represent agreement as it is being developed. In terms of the second problem, we should be able to automate the process of incorporating new meanings. But to deliver on this potential will require research progress.

3 ONTOLOGIES: WHERE'S THE RESEARCH?

We need to concentrate some research on the representation of agreements. That is, we must be able to construct knowledge bases that clearly delineate the agreed-upon aspects of concepts, and that specifically support the partial nature of agreement knowledge. Different kinds of knowledge (e.g., symbolic definitions, assertions supported by a set of data, probability-based characterizations) must be tied to a central "core of agreement": it must be clear to agent developers and interacting agents what has been agreed upon and how that agreement has been expressed.

We must also do research on treating ontologies as ever-evolving bodies of knowledge. The key is to allow ontologies to grow, but to preserve their underpinning set of agreements — and to know when (and how) these agreements need to be changed. Usually evolution will come through the need to incorporate recently understood ramifications. The research problem is to automatically determine when these ramifications require agreements to change (i.e., when they go beyond specialization of the existing representations) and then to demonstrate what changes are required to accommodate the new ramifications.

Another important kind of meaning evolution — indeed the Holy Grail of knowledge base design — is using an existing set of concepts to support an unplanned set of tasks. Here the issue is often in understanding what elaboration will be required to incorporate the new uses for the ontology. This is a major research issue, but I think that a possible way to bound the problem is to define it in terms of an enlarged set of interactions that must be supported. That is, if the ontology is originally defined in terms of the set of inter-agent interactions it must support, it might be possible to definitively determine the differences that are introduced by a new set, and how the ontology must be changed to support them.

PANEL

Implementations and Research: Discussions at the Boundary

Robert MacGregor
Chair

James Hendler **Stuart Shapiro**
Lenhart Schubert **Howard Shrobe**

Abstract

The amount of transfer between theory and practice in the field of knowledge representation (KR) is open to debate. One measure of the utility of theoretical results in KR is the degree to which these results have influenced the design of implemented KR systems. Each of the panelists has participated in the construction of a significant knowledge representation system. We have asked them to address the following questions:

1. How have you, as an implementer of a KRS (KR system), benefitted from existing theoretical and/or other published results in AI?
2. What future research directions would you like to see, i.e., what kinds of research would benefit future KRSs?
3. Where are KRSs headed? What new kinds of inference capabilities do you hope/expect to see implemented? Are general-purpose KRSs a viable concept, i.e., can they compete successfully with niche KRSs?

Panelist's position papers follow.

PANEL POSITION STATEMENT

Implementations and Research: Discussions at the Boundary

Robert M. MacGregor
University of Southern California, Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292, USA
macgregor@isi.edu

The Loom team at USC's Information Sciences Institute has developed Loom, a KR system in use worldwide, and is working on a follow-on called PowerLoom. Our goal is to produce a system that is representationally expressive, provides a variety of efficient inference modes, and integrates well with complimentary commercial tools and standards (e.g., World Wide Web, CORBA, and RDBMSs).

A KR system designer must (i) define a semantics for the system's representation language, and (ii) invent an architecture for representing and reasoning in accordance with that semantics. The Loom systems are logic-based—the major semantic decisions involve extensions: knowledge base revision; default logic; reasoning over an inconsistent knowledge base; contexts; temporal logic; and production-rule triggering.

In the area of semantics, the most valuable contributions from the theoretical literature are (i) to provide a vocabulary for conceptualizing semantic issues (e.g., "stratified", "skeptical", "shortest-path", "possible worlds", "analytic", "decidable"), and (ii) to provide evaluation criteria for accepting or rejecting various semantic possibilities. Specific logics proposed in the literature seldom bridge the gap between theory and practice. In no case did we adopt a published logic specification wholesale into Loom; we always found it desirable to modify an existing proposal, or to invent our own semantics. For example, in the area of belief revision, Loom adopts a (semantically-based) mechanism called "clipping" derived from a similar notion in the FORBIN system. We consider syntactically-based notions such as those based on "minimal-mutilation" to be completely unsuitable. For default reasoning, we consider the complexity of skeptical inheritance to be unacceptably high ("polynomial-time" does not equate with "efficient"). Loom adopts a more restrictive scheme for handling conflicts that is simpler and faster.

We find the theoretical literature less useful (with respect to our own research) when it comes to architectural issues. For an expressive KR system, complexity issues are irrelevant, except as a means to justify incomplete reasoning. Most of the published algorithms apply only to restricted logics (e.g., the propositional calculus, Brand X temporal logic, Brand Y description logic) and may find utility only in niche systems (e.g., a CSP solver, a scheduler, or a configurator). Our most valuable sources of architectural ideas have come from conversing with other system implementers and from reading system source code (NIKL, SCREAMER, and O-PLAN).

One of the principal challenges in the Loom systems is finding an acceptable means for characterizing incomplete reasoning. For PowerLoom, we plan to establish "islands of completeness" within an otherwise incomplete system. For example, PowerLoom will be able to reason completely with a Horn-rule knowledge base and with a propositional knowledge base. It may also provide complete inference for some description logics. Users will be able to add their own rules of inference to correct "holes" in the built-in inference scheme. However, we are finding that writing rules to reason about negation or disjointness is difficult—we would like to see more work in this area.

We see a tension in the AI marketplace between general-purpose KR systems like Loom, and the niche systems. For any application for which a niche system is applicable, the niche system can be expected to outperform a more generic system. On the other hand, "AI-complete" applications such as natural language processing seem to demand generality. We expect the emergence of large-scale ontologies to provide another field where expressiveness and generality are prerequisites.

We would like to see theoretical advances in the following areas:

1. Reasoning in the presence of inconsistency—we expect that large knowledge bases will inevitably contain inconsistencies;
2. Characterizations of incomplete reasoning and/or identification of sublogics contributing to new "is-

lands of completeness";

3. Logics of abstraction hierarchies;
4. Combining classical logical deduction with probabilistic reasoning;
5. Other hybrid logics that combine classical logic with something else.

PANEL POSITION STATEMENT

Implementations and Research: Discussions at the Boundary

James Hendler
Computer Science Department
Institute for Systems Research and Institute for Advanced Computer Studies
University of Maryland, College Park, MD 20742, USA
hendler@cs.umd.edu

Before I answer the questions addressed to the panel, I wish to make it clear that I think that absolutely the most critical issue for KR systems of the future is scaling to very large knowledge bases. As such, my answers are driven by the need to develop better algorithms and systems, not by the desire for ever more expressive languages that can only handle toy problems. To put my money where my mouth is, my research group's KR System, PARKA, works with knowledge bases with millions of assertions in interactive applications needing rapid responses. My answers are motivated by my need to push this still larger and faster in response to the requirements of the information technology and datamining applications my students and I are building.

(1) How have you, as an implementer of a KRS (KR system), benefitted from existing theoretical and/or other published results in AI?

Most of the work in KR has focused on improved expressivity. Unfortunately, as all KR implementors are aware of, there is usually a tradeoff between expressivity and efficiency. Thus, much of the work has been useless as it has been unimplementable. One exception to this is the work in inheritance, which includes some significant algorithmic work (the work leading to the IDO algorithm for skeptical inheritance). By exploring the problems with these algorithms, my group was able to find polynomial algorithms for inheritance that allow a large class of exceptions to be handled. This has allowed us to gain a lot of expressivity (the exceptions and exception handling) without giving up efficiency (the polynomial nature of our algorithm). As the AI community learned many years ago (although many of us seem to have forgotten it), inheritance is the basis for most of the needs of KR users, and is one of the primary things we can bring to the table which other communities (especially DB researchers) don't.

(2) What future research directions would you like to see, i.e., what kinds of research would benefit future KRSs?

In looking at expressivity vs. efficiency, it is instructive to look at the other end of the spectrum from KBs. The database community has largely focused on efficiency and sacrificed the expressivity of KR Systems. The result? They outsell us by several orders of magnitude!!! We must start looking more seriously at intermediate points in the spectrum and develop systems which are more expressive than DBs without sacrificing efficiency. PARKA, for example, can now support complex inheritance, recognition queries, structure matching, and even prolog-like logic on very large KBs on single processors in times sufficient for the interactive needs of many applications.

What is the magic behind Parka's performance? Answer: a lot of use of database technology (where appropriate) coupled with a lot of concentration put into our algorithms. The system we have now is efficient on PCs and workstations and efficiently parallizable on the fastest of supercomputers. In addition, we have recently shown that the key algorithms will scale well on distributed systems (including fast networks of PCs).

I believe that there is fundamental work yet to be done on exploring the tradeoffs between space and time that separate knowledge representation systems from databases. There are some exciting applications in datamining, case-based reasoning, and scientific support systems, to name just a few, which need knowledge-based expressivity at data-based speeds and sizes. The AI/KR community must start looking at these issues and claiming them for AI. Otherwise, the DB folks will get there first (Ullman recently gave an invited talk at the AAAI conference. Get my point?)

(3) Where are KRSs headed? What new kinds of inference capabilities do you hope/expect to see implemented? Are general-purpose KRSs a viable concept, i.e., can they compete successfully with niche KRSs?

It should be clear from the above that I think KRSs are actually heading "back down" a bit. We must temporarily move away from the full richness of higher-ordered logics and non-parallelizable exponential algorithms, and move towards systems that actually work

and do things for people.

There is a great need to expand the range of capabilities of such systems - handling temporal inferences, context mechanisms, etc. The theoretical research in such systems is out there, let's now look at how to make these things really work! The middle of the expressivity is largely unexplored, and my group is enjoying a lot of visibility right now for being the first ones there. However, there's a lot of room for others, and I hope the community will realize the importance of this "nuts and bolts" work (as the reviewer who rejected my paper for KR so eloquently put it).

Finally, as to the question of "general-purpose KRSs", I'm a firm believer in these and think their time is coming. We can't beat niche KRSs in their niches, but damned if we can't do a lot more than any niche language across a wide spectrum. The DB community has lots of general purpose system that can't outperform special purpose DBs, but they sure as heck outsell them!! The situation in AI should be the same – general purpose KRSs, implemented efficiently enough to do well across a wide range of tasks are clearly doable – let's get off our butts and get them built!

PANEL POSITION STATEMENT

Implementations and Research: Discussions at the Boundary

Lenhart Schubert
Computer Science Department University of Rochester Rochester, NY 14627, USA
schubert@cs.rochester.edu

1 The EPILOG System

EPILOG is the computational system developed for Episodic Logic (EL), a very expressive, NL-like logic. Its goals are to support NLU and commonsense reasoning using large KBs of commonsense knowledge. EPILOG has been under development at U of Alberta and U of Rochester (with Boeing support '87-'92) for over 20 years. Previous versions were called ECONET and ECOLOGIC; EPILOG was first delivered to Boeing in 1990. It is fully implemented, and has been demo'd for story fragments, aircraft maintenance reports, and the notorious Steamroller theorem proving problem. EPILOG will shortly become available by anonymous ftp to cs.rochester.edu, directory pub/packages/knowledge-tools. A web page is in the works. A stand-alone, new version of the time specialist can be obtained by anonymous ftp to cs.rochester.edu, directory pub/packages/knowledge-tools, files tg-ii.readme and tg-ii-1.tar.gz. (This is a scalable inference system, to our knowledge the fastest of its kind.)

The knowledge representation, EL, is "natural" in the sense that it allows direct expression of many NL constructs. This makes it relatively easy to map NL to EL and vice versa, to browse a KB, and to formulate commonsense inferences. More particularly, EL is a first-order intensional logic featuring

- explicit episodes (situations, events) associated with arbitrary sentences (wffs)
- lambda abstraction
- sentence and predicate modifiers (sim. to adjectives, adverbials)
- sentence and predicate nominalization (reification) operators
- a DRT/DPL-like mechanism for making referential connections
- statistical conditionals, and epistemic probabilities

EPILOG uses input-driven inference for understanding and goal-driven inference for question answering and problem solving. Both modes are based chiefly on replacing positively embedded subformulas by their consequences, and negatively embedded formulas by their anticonsequences (this subsumes resolution, but is not based on skolemization). Forward inference termination is by probability threshold and "interestingness". General inference is supported by efficient, uniformly integrated specialists for time, taxonomies, parts, colors, episodes, sets, numbers, strings, schematic axioms, etc. These perform simplification, factoring, and the equivalent of narrow theory resolution. The key to scalable knowledge access and inference is an indexing scheme based on <topic (predicate), participant type, role> triples, plus mechanisms for "hierarchy climbing" and modal embedding.

2 Theoretical influences

The KRS design was inspired originally by the semantic net literature (Quillian, Shapiro, Rumelhart, etc.) and later by Montague-style logical form, Turner/Chierchia type theory, Barwise/Perry situation theory, and Kamp/Heim discourse representation theory, among others. The inference techniques were originally resolution-based, but were subsequently influenced by natural deduction techniques (e.g., a la F.J. Pelletier). However, our style of "polarity-based" inference could be said to hark back to pre-Fregean "natural logic" – see van Benthem's discussion in ch.6 of his Essays in Logical Semantics).

The integration of the specialists benefited from early work on procedural attachment, and later from Stickel's notion of theory resolution. Probability chaining is based on work on probabilistic logics (Bacchus, Halpern, etc.); evidence combination is Bayesian at times, and more ad hoc at other times. The knowledge accessing structure was inspired by the semantic net literature; the notion of "topic hierarchies" used in accessing turns out to be similar to Pustejowsky's more recent notion of "qualia".

3 Needed research

Some of the most urgently needed research concerns the interface between KRSs and the user/world. A KRS, no matter how elegant and powerful its mechanisms may be, is useless if it is hard to build up a substantial knowledge base, and hard to communicate with the system.

Thus we need research on how KRSs can "learn by being told" (and by making generalizations, etc.), so that we can begin to break through the knowledge bottleneck. And we need research on more effective, natural interaction with KRSs. Both goals require good ways of transducing between the KR and ordinary language (as well as other media – graphics, menus, gestures, etc.)

In representation proper, we need further development of the model theories and proof theories of highly expressive logics such as EL, as a basis for well-founded implementations. Weak representations with efficient proof methods are of interest as a basis for "subroutines", but not as a basis for stand-alone systems.

Important research topics in inference include

1. better-founded, more complete, incremental methods of "adjudicative inference": computing propositional probabilities or probability bounds for arbitrary wffs in the light of various lines of uncertain inference supporting that wff, or its negation;
2. more general, more effective methods of PLANNING (and plan recognition); planning should be probabilistic and incremental. This is crucial not only for systems specifically designed to help with planning, but ANY intelligent system that can communicate naturally with a user;
3. better-founded, more complete methods of predicting propositional attitudes (beliefs, plans, desires, etc.) of other agents who "think like us" based on what we know they know, and on what we know they find out (through perception, being told, etc.);
4. further development of "general-purpose specialists" (e.g., for envisioning dynamic, interacting objects) and integration with general reasoners; the completeness problems that arise for hybrid systems are very hard.

4 Prospects

KRSs seem headed for greater expressiveness and generality, coming closer to matching the resources of NL. In part this is driven by the growing emphasis on sharing ontologies and KBs. It is also driven by the growing emphasis on access to KBs via natural, discourse-like interaction.

The most important development goals for EPILOG, over the next few years, are (a) the addition of incremental, probabilistic planning capabilities, based in part on stored plans that are represented and indexed much like facts (with goals providing indexing keys), and on methods for synthesizing, modifying, and combining plans; (b) developing more complete, better-founded (Bayes-like, or entropy-maximizing) probabilistic inference methods; and (c) building sufficiently large core KBs to allow knowledge bootstrapping via NL.

I see general-purpose KRSs as providing the "commonsense core" of various application systems, containing the knowledge essential to natural interaction with computers, regardless of the application. The core KRS could mediate access to many special-purpose applications packages, based on knowledge about the functionality of these packages and about the goals of the user.

PANEL POSITION STATEMENT

Implementations and Research: Discussions at the Boundary

Stuart C. Shapiro
Department of Computer Science and Center for Cognitive Science
State University of New York at Buffalo
Buffalo, NY 14260-2000, U.S.A.
shapiro@cs.buffalo.edu

I will respond to the questions posed to the panel in three parts: niche vs. general-purpose KRSs; the past; the future.

1 Niche vs. General-Purpose KRSs

At the most basic level, a KRS is a system that stores data and computes new data from what is stored. Almost any computer program satisfies that description. So I conclude that a KRS for a narrow enough domain D is just a program for D, and, however valuable, is neither a KRS nor even an AI system, although it might be a useful subroutine for a KRS.

At the other extreme, completely general KRSs merge imperceptibly with DBMSs, theorem provers, and logic programming languages. Researchers and implementors in these four fields should, indeed, interact with each other even more than they do now, but I believe that the field of KR has a niche that distinguishes it from the other three.

I believe that the defining characteristic of a KRS is that it should have human-level abilities. I, personally, do not care if a KRS exactly mimics the human mind, to the extent, for example, of making the same mistakes people tend to make, but I would want its behavior to be within the human range in terms of ability, generality, and ease of "use." The touchstone of "ease of use" is the ability to take part in a conversation in natural language, including the natural accompaniments such as gestures, and drawing diagrams. Therefore, I believe that the right level in the special purpose–general purpose spectrum is the level to support I/O in a human natural language. This is very general, yet still puts severe constraints on the KRS.

2 The Past

SNePS, the KRS my colleagues, students and I have been responsible for, is and has been an implementation of our own theories of KR and natural language competence, which we have published in an extensive series of papers. So one answer to the question of how we have "benefitted from existing theoretical and/or other published results in AI" is that the implementation of SNePS has always been carried out in close coordination with the development of our own theoretical and/or published results. We have also kept abreast of the theoretical and/or published results of others, and adapted or built on their work when we felt that was appropriate. One significant example of this is that the unification algorithm currently used in SNePS 2 is a direct adaptation of the one described in [1].

3 The Future

As I indicated above, KRS implementors can benefit from the work in DBMSs, theorem provers, and logic programming languages. There has been some interaction among these communities, but I think we could benefit from even more. For example, a KRS may be viewed as a logic programming language programmed by the informant in NL which is then translated into the LP language. We probably need some of the LPL researchers to look at some of the more useful logics developed for KR, and to develop efficient techniques for interpreting or incrementally compiling the LP languages, rather than just at compilation techniques, so that the input-query cycle is fast, and so that enough of the original input is preserved to provide data for explanation in NL.

I think more work needs to be done on using negation in KRSs. We have known about "the problem of fast negation" since Collins and Quillian, but it has usually been swept under the rug of "negation by failure." Recently, some good work has been done on negation, but more needs to be.

I think we need to look more at incorporating constraint satisfaction techniques in KRSs. Constraint satisfaction has proved useful in expert systems, and logic programming people are looking at it. It is proba-

bly a useful additional reasoning mechanism for KRSs.

I think we need to do more with reasoning about propositions about sets and collections, which derive from NL sentences containing plurals.

Finally, I think we need to look at how KRSs can derive pragmatically correct answers to questions. People do not just answer questions with "logically correct" answers, but with pragmatically appropriately answers. Currently this is viewed as a stage above that of the operation of the KRS, but perhaps it could be pushed down into it.

References

[1] G. Escalada-Imaz and M. Ghallab, "A Practically Efficient and Almost Linear Unification Algorithm," *Artificial Intelligence* , 2 (Sept. 1988), 249-263.

Related Conferences

KR Conference Coordination*

Abstract

The following conferences, symposium, and workshops are detailed:

1. TARK VI: March, 1996, Renesee.
Johan van Benthem, Yoav Shoham

2. NONMON VI: June, 1996, Timberline.
Moises Goldszmidt, Vladimir Lifschitz

3. FAPR: June, 1996, Bonn.
Dov Gabbay

4. CD II: June, 1996, Bonn.
Thomas Gordon

5. DL: November, 1996, Boston.
Lin Padgham

6. KRNL: November, 1996, Boston.
Lucja Iwanska

1 TARK VI

Theoretical Aspects of Reasoning about Knowledge

According to the call:

> The mission of the bi-annual TARK conferences is to bring together researchers from a wide variety of fields - including Artificial Intelligence, Cryptography, Distributed Computing, Economics and Game Theory, Linguistics, Philosophy, and Psychology - in order to further our understanding of interdisciplinary issues involving formal reasoning about rationality and knowledge. Topics of interest include, but are not limited to, semantic models for knowledge, for belief, and for uncertainty, bounded rationality and resource-bounded reasoning, commonsense epistemic reasoning, knowledge and action, applications of reasoning about knowledge and other mental states, and belief revision.

The invited speakers, invited tutorials, and invited talks were:

> **Gardenfors**, Belief Revision and Knowledge Representation
>
> **Kalai**, Rational Interactive Learning in Economics and Game Theory
>
> **Papadimitriou**, Games, Information, and Computational Complexity
>
> **Pearl**, Causality, Counterfactuals and Implicit Actions
>
> **Rubinstein**, Imperfect Recall in Decision Problems
>
> **Sundholm**, Constructive Proof Theory and Epistemics
>
> **Veltman**, Dynamic Update Semantics

The papers were:

> **Aumann, Hart, Perry**, The Absent-Minded Driver
>
> **Boutilier**, From reinforcement learning to emergent conventions
>
> **Fagin, Halpern, Moses**, Special Session: Common Knowledge Revisited in Computer Science

*Prepared by R. Loui with help from B. Verheij.

Farrinas del Cerro, Herzig, Belief Change and Dependence

Flach, Rationality Postulates for Induction

Halpern and Lakemeyer, Multi-Agent 'Only Knowing'

Halpern, Time consistency and Strategy in Games of Imperfect Recall

Kfir-Dahav, Tennenholtz, Multi-Agent Belief Revision

Minelli, Polemarchakis, Knowledge at Equilibrium

Morris, Special Session: Common Knowledge Revisited in Economics and Game Theory

Nayak, Foo, Pagnucc, Sattar, Changing Conditional Beliefs Unconditionally

Ramanujam, Local Knowledge Assertions in a Changing World

Rodrigues, Ryan, Schobbens, Counterfactuals and Updates as Inverse Modalities

Schlechta, Lehmann, Magidor, Distance Semantics for Belief Revision

Thomason and Horty, Nondeterministic Action and Dominance: Foundations for Planning and Qualitative Decision

van der Meyden, Knowledge-Based Programs: On the Complexity of Perfect Recall in Finite Environments

Vardi, Implementing Knowledge-Based Programs

Vardi, Review of knowledge-based programs

2 NONMON VI

Workshop on Nonmonotonic Reasoning

According to the call:

> The aim of the workshop [was] 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. The field of nonmonotonic reasoning includes work on circumscription, autoepistemic and default logic, truth maintenance, closed-world databases, logic programming, probabilistic reasoning, and related systems.

Additionally:

> The theory of nonmonotonic reasoning has helped provide a clear and formal framework that can be used to understand and compare issues in action representation, planning, and other areas. This workshop will emphasize pushing the state of the art, to ensure that focus is directed towards the applications of these notions in the current problems being addressed by AI researchers.

Some organizing themes:

> Innovative approaches
> Implementation and practical issues
> Describing applications
> Theoretical foundations
> Relations between different approaches

The papers presented were:

Amati, Aiello and Pirri, Modal nonmonotonic reasoning via boxed fixpoints

Aytekin-Kurban and Kurtz, Feasible commitment in predicate circumscription

Bamber, A doubly probabilistic approach to a nonmonotonic logic of generalizations

Baral, Gabaldon and Provetti, Value minimization in circumscription

Barwise, State spaces and nonmonotonicity

Boutilier and Goldszmidt, The frame problem and Bayesian network action representation

Brewka, A framework for well-founded default logic

Delgrande, Towards a first-order conditional logic for default properties

Engelfriet, Minimal temporal epistemic logic

Froehlich and Nejdl, A model-based reasoning approach to circumscription

Ginsberg, Do computers need common sense?

Grosof, Practical prioritization for logic programs

Lehmann, The structure of subjective plausibility

Lobo and Uzcategui, Abduction and change

Risch, Cumulative default theories vs noncumulative default logic

Suchenek, Indefinite models and parallel positive resolution for disjunctive stratified logic programs

Zaverucha, On cumulative default logics with filters

The panels were:

Do computers need common sense?

Implementations of nonmonotonic reasoning systems

The future of nonmonotonic reasoning as a subfield of AI

3 FAPR

Formal and Applied Practical Reasoning

According to Gabbay and Ohlbach (eds.), in *Practical Reasoning*, Springer, 1996:

> The main purpose of ... FAPR was to ... introduce [AI, philosophy, psychology, and linguistics] to each other, to compare the current state of research in these areas, and to make such research available to all researchers involved.

The main idea:

> Significant communities of researchers, faced with the shortcomings of traditional deductive logic in modelling human reasoning and argumentation, are actively engaged in developing new approaches to logic (informal logic, dialogue logic) and argumentation (rhetoric, pragmatic and dialectical) that are better suited to the task.

The invited talks were:

Hintikka, Logic as a Key to Good Reasoning

Reiter, Cognitive Robotics

Walton, How Can Logic Best Be Applied to Arguments?

Zadeh, The Role of Fuzzy Logic in Commonsense Reasoning and Knowledge Representation

The workshops were:

Cunningham and Caferra, Medlar Workshop on Formal and Informal Reasoning

Dubois, Farinas del Cerro, Herzig and Prade, Is Logic + Ordering Enough for Formalising Common Sense Reasoning?

Gordon and Brewka, Computational Dialectics: Models of Argumentation, Negotiation and Decision Making

Sigmund and Thielscher, Reasoning about Actions and Planning in Complex Environments

The tutorials were:

Blair, Evaluating Arguments

Cohn, The Challenge of Qualitative Spatial Reasoning

Demolombe and Jones, Modal-logical Specification of Security Requirements

Geffner, Default Reasoning

Gabbay, Elements of Labelled Deductive Systems

Jameson, Perspectives on Belief Ascription

Stilman, Linguistic Geometry: A Formal Model of Human Reasoning for Solving Search Problems

The organization was:

User Modelling and Belief
Legal Reasoning
Reasoning about Real Systems
Miscellaneous Reasoning 1
Argumentation 1
Philosophical Aspects
Default Reasoning 1
Analogy
Calculi
Default Reasoning 2
Reasoning and Change
Temporal and Procedural Reasoning
Modal Logics and Metareasoning
Actions and Agents
Miscellaneous Reasoning 2
System Descriptions
Argumentation 2

The papers were:

Antoniou, A Comparative Survey of Default Logic Variants

Balbiani, Farinas del Cerro, Tinchev and Vakarelov, Geometrical Structures and Modal Logic

Balbiani, Modal Logics with Relative Accessibility Relations

Cimatti and Serafini, Mechanizing Multi-Agent Reasoning with Belief Contexts

Das, Fox and Krause, A Unified Framework for Hypothetical and Practical Reasoning(1) - Theoretical Foundations

Doherty, Lukaszewicz and Szalas, General Domain Circumscription and its First-Order Reduction

Duc, Reasoning about Rational, but not Logically Omniscient Agents

Engelfriet and Treur, Specification of Nonmonotonic Reasoning

Engelfriet, Herre and Treur, Nonmonotonic Reasoning with Multiple Belief Sets

Errico and Aiello, Intelligent Agents in the Situation Calculus: an Application to User Modelling

Farinas del Cerro, Frias Delgado and Herzig, Talkin'bout Consistency, or: when Logically Possible becomes Possible

Feteris, The Analysis and Evaluation of Legal Argumentation from a Pragma-Dialectical Perspective

Finocchiaro, Reasoning about Reasoning

Fisher, Wooldridge, Dixon, A Resolution-Based Proof Method for Temporal Logics of Knowledge and Belief

Fox and Das, A Unified Framework for Hypothetical and Practical Reasoning(2) - Lessons from Medical Applications

Garigliano, Type Theoretic Semantics for Sem-Net Simon Shiu, Zhaohui Luo and

Geffner, A Formal Framework for Causal Modeling and Argumentation

Gilbert, Goals in Argumentation

Giordano, Martelli and Sapino, An Abductive Proof Procedure for Conditional Logic Programming

Girle, Commands in Dialogue Logic

Giunchiglia and Giunchiglia, Ideal and Real Belief about Belief

Gotoda, An and Fujiwara, Analogical Reasoning of Organic Reactions Based on the Structurized Compound-Reaction Diagram

Governatori, Labelling Ideality and Subideality

Green, Arguments and Mental Models: a Position Paper

Grootendorst, The Future of Argumentation Theory

Grste, Herrmann and Sandner, SEdit - Graphically Validating Technical Systems

Guarini, Mind, Morals, and Reasons

Hansen, Aristotle, Whately, and Taxonomy of Fallacies

Johnson, The Need for a Dialectical Tier in Arguments

Kerber and Melis, Two Kinds of Non-Monotonic Analogical Inference

Kloosterhuis, The Normative Reconstruction of Analogy Argumentation in Judical Decisions: a Pragma-Dialectical Perspective

Kreitz, Lau and Ornaghi, Formal Reasoning about Modules, Reuse, and their Correctness

May and Schmitt, A Tableau Calculus for First-Order Branching Time Logic

Meyer and Leeuwen, Possible World Semantics for Analogous Reasoning

Montesi, Renso and Turini, Using Temporary Integrity Constraints to Optimize Databases

Neumann, Graded Inheritance Nets for Knowledge Representation

Oliveira, The Implementation of CondLP

Oliveira, The Implementation of LENA

Omelyanchyk, How to Reason About Akratic Action Practically?

Parsons and Fox, Argumentation and Decision making: a Position Paper

Parsons, Defining Normative Systems for Qualitative Argumentation

Plug, Complex Argumentation in Judicial Decisions: Analysing Conflicting Arguments

Pohl, Combining Partitions and Modal Logic for User Modeling

Pollock, Reason in a Changing World

Prakken and Sartor, A System for Defeasible Argumentation, with Defeasible Priorities

Prendinger, Modal Logic for Modelling Actions and Agents

Radzikowska, Formalization of Reasoning about Default Action

Reed, Long and Fox, An Architecture for Argumentative Dialogue Planning

Rodrigues and Gabbay, AA Methodology for Iterated Theory Change

Schaub and Thielscher, Skeptical Query-Answering in Constrained Default Logic

Tindale, From Syllogisms to Audiences: The Prospects for Logic in a Rhetorical Model of Argumentation

Van der Acker and Vanthienen, Integrating Statistical Audit Evidence with Belief Function Theory

Vogel, Human Reasoning with Negative Defaults

Weber, On the Semantics of the Unknown

Weydert, System J - Revision Entailment - Default Reasoning through Ranking Measure Updates

Woods, Deep Disagreements and Public Demoralization

Wooldridge, Practical Reasoning with Procedural Knowledge

Yi, Towards the Assessment of Logics for Concurrent Actions

4 CD II

Computational Dialectics Workshop

According to the call:

> The workshop [discussed] formal models of argumentation and negotiation as well as approaches that model group decision making processes. In each case it is necessary to weigh reasons for and against a certain option. Both symbolic and quantitative approaches [were] welcome.

Additionally,

> The term "Computational Dialectics" describes an area of Artificial Intelligence which investigates computational models of the processes by which groups of natural or artificial agents construct judgement, agreement, or other forms of social choice. Special emphasis lies on methods for recognizing or achieving an outcome in a fair and effective way.

The papers were:

Bonet and Geffner, Arguing for Decisions: A Qualitative Model of Decision Making

Gordon and Karacapilidis, The Zeno Argumentation Framework

Krabbe, Critical Discussion and the Problem of Retraction

Pollock, The structure of defeasible argumentation

Prakken, Dialectical proof theory for defeasible argumentation with defeasible priorities

Prakken and Sartor, Representing legal precedents as defeasible–argumentation structures

Reed and Long, Argumentation as Intention-Laden Persuasive Communication

Verheij, Two Approaches to Dialectical Argumentation: Admissible Sets and Argumentation Stages

Vreeswijk, Representations of Formal Dispute with a Standing Order

5 DL

International Workshop on Description Logics

Having just concluded at the time of this conference, DL is:

> devoted to discussing developments and applications of description-oriented knowledge representation formalisms based on logic.

The main themes were to be:

> Foundations Extensions
> Integration with other formalisms
> Specification
> Applications

The papers were to be:

Artale, Franconi, Guarino, Open Problems with Part-Whole Relations

Baader and Sattler, Knowledge representation in process engineering

Beeri, Levy and Rousset, Applying Description Logics to Database Problems: Position Statement

Bermudez De Andres, Illarramendi, Blanco, Goni, Incorporating New Languages into DL Systems

Borgida, Isbell and McGuinness, Reasoning with Black Boxes: Handling Test Concepts in CLASSIC

Buehrer, Liu, Hong, Class Algebra as a Description Logic

Cadoli, Lenzerini, Palopoli, Datalog and Description Logics: Expressive Power

Calvanese, De Giacomo and Lenzerini, Representing SGML Documents in Description Logics

De Giacomo, Donini, Massacci, Exptime Tableaux for ALC

Elhaik, Rousset, Gaudel, A Proposal for a Glass-Box Approach for Subsumption Checking

Gil and Gonzalez, Subsumption-Based Matching: Bringing Semantics to Goals

Giunchiglia, Roveri and Sebastiani, A new method for testing decision procedures in modal and terminological logics

Haarslev, Using Description Logic for Reasoning about Diagrammatical Notations

Haraguchi, Towards a Legal Reasoning System based on Description Logics

Horrocks and Rector, Using a Description Logic with Concept Inclusions

Kamp and Wache, Using description logics for consistency-based diagnosis

Kudenko and Hirsh, Representing Sequences in Description Logics

Lambrix and Padgham, A Description Logic for Composite Objects for Domain Modeling in an Agent-Oriented Application

Larab and Benharkat, Terminological Model: Correspondence Refinement

Latourrette, Simonet, An Approach to Subsumption in a DL with Implication

Liebig, Roesner, Extending Description Logic for the Modelling of Technical Objects

Mihoubi, Simonet, Simonet, A Translation Scheme for Domain Ontologies Based on Model Ontologies of KBS

Moeller, Integration of Description Logics with other formalisms, such as object-oriented languages

Napoli, A Proposal for a Layered Architecture for a Hybrid Object-Based Representation System

Nijhuis, The use of Description Logics in the Condorcet conceptual information retrieval system

Oles, Mays, Weida, The Algebraic Essence of K-Rep

Rosati, On the semantics of Epistemic Description Logics

Speel, Can Description Logics be used in Real-Life Knowledge-Based Systems?

Ventos, A deductive study of the C-CLASSIC Description Logic

Weida and Mays, The K-Rep System Architecture

6 KRNL

Symposium: Knowledge Representation Systems Based on Natural Language

This symposium follows KR as a part of the AAAI Fall Symposium Series. According to the call:

> The Symposium addresses the theoretically and practically important problem of knowledge representation (KR) systems that closely parallel the representational and inferential characteristics of natural language (NL). Advantages of such NL-based KR systems would be enormous.
>
> ... The goal of this symposium is to address in-depth such arguments for and against designing KR systems closely simulating natural language.

Additionally,

> Many philosophers, linguists and cognitive scientists believe that mental-level representation of knowledge (human mind) is close in form to natural language. While some AI researchers believe that it is feasible and necessary to design KR systems closely mimicking natural language, others are pessimistic about success or even possibility of designing such KR systems. This pessimism might account for the general lack of interest in the problems of NLP within the KR community.

The list of papers was not available at the time of this writing.

Author Index