Commonsense 2009

Commonsense 2009

Proceedings of the Ninth International Symposium on Logical Formalizations of Commonsense Reasoning

June 1-3, 2009

The Fields Institute
on the campus of the University of Toronto

Edited by
Gerhard Lakemeyer
Leora Morgenstern
Mary-Anne Williams

ISBN 978-0-9802840-6-5
26 May 2009

Published by UTSePress
University of Technology, Sydney
Broadway, NSW 2007 Australia

Table of Contents

Technical Papers

Preface

The Ninth International Symposium on Logical Formalizations of Commonsense Reasoning will be held at the Fields Institute for Research in Mathematical Sciences, at the University of Toronto, on June 1–3, 2009. Since its inception in 1991, the Commonsense Reasoning Symposium series has provided a forum for exploring one of the long-term goals of Artificial Intelligence, endowing computers with common sense. Although we know how to build programs that excel at certain bounded or mechanical tasks which humans find difficult, such as playing chess, we still have very little idea how to program computers to do well at commonsense tasks which are easy for humans. One approach to this problem is to formalize commonsense reasoning using formal languages such as mathematical logic. The focus of the symposium is on representation rather than on algorithms, and on formal rather than informal methods.

Twenty-two technical papers, on a variety of topics in commonsense reasoning, including physical reasoning, planning, theories of action, belief revision, and nonmonotonic reasoning, are included in these proceedings and will be presented at the symposium. Each paper was reviewed by at least two members of the program committee.

The program also features invited talks by three leading researchers:

- Anthony G. Cohn (University of Leeds, UK), “Acquiring Commonsense Knowledge from Perceptual Observation”;
- Ernest Davis (New York University, USA), “Commonsense Reasoning about Chemistry Experiments: Ontology and Representation”; and
- Sheila McIlraith (University of Toronto, Canada), “Diagnosis Revisited.”

We are pleased to recognize two student papers with the Commonsense-2009 Outstanding Student Paper Award. Eligible papers had to be authored or co-authored by a student at the time of submission, and could not be co-authored by any of the Symposium Chairs. The two Outstanding Student Papers are:

- Shakhil Khan and Yves Lesperance: “A Logical Account of Prioritized Goals and Their Dynamics”
- Hannes Strass and Michael Thielscher: “Defaults in Action: Nonmonotonic Reasoning about States in Action Calculi”

We are very pleased that the symposium will be held in Toronto this year, where Ray Reiter (1939–2002), a world leader in cognitive robotics and formal commonsense reasoning, spent many years of his life and scientific career. It is with great gratitude for Ray’s lasting contributions to our field that we dedicate this symposium to his memory.

Organizing such an event always rests on many shoulders. We are especially grateful to Hojjat Ghaderi, our Local Arrangements Chair, as well as the other members of his team from the University of Toronto: Luna Keshwah, Hector Levesque, and Sheila McIlraith. We are equally grateful to our Conference Webmaster, Benjamin Johnston, of the University of Technology, Sydney, for designing and building the symposium website, managing the EasyChair conference system, and preparing these proceedings.

The Fields Institute for Research in Mathematical Sciences has been extraordinarily generous, in providing us the space for the symposium; providing funds for student and symposium chair travel; managing registration; and publicizing this event. We especially

thank Alison Conway of the Fields Institute for her help in organizing all of these functions.

We also thank the Centre for Quantum Computation and Intelligent Systems at the University of Technology, Sydney, for its generous support of student travel; and the IBM T.J. Watson Research Center for partly subsidizing the travel of one of the winners of the Commonsense-2009 Outstanding Student Paper Award.

This symposium is held in cooperation with AAAI, the Association for the Advancement of Artificial Intelligence. We thank them for their help in promoting this symposium.

Aachen, New York, and Sydney, May 2009
Gerhard Lakemeyer
Leora Morgenstern
Mary-Anne Williams

Organization

Program Chairs

Gerhard Lakemeyer, RWTH Aachen University, Germany
Leora Morgenstern, New York University, New York, USA
Mary-Anne Williams, University of Technology, Sydney, Australia

Program Committee

Eyal Amir, University of Illinois at Urbana-Champaign, USA
Chitta Baral, Arizona State University, USA
Johan van Benthem, University of Amsterdam, the Netherlands
Xiaoping Chen, University of Science and Technology of China, China
Ernest Davis, Courant Institute, New York University, USA
Patrick Doherty, Linkoping University, Sweden
Esra Erdem, Sabanci University, Turkey
Norman Foo, University of New South Wales, Australia
Alfredo Gabaldon, University of New South Wales, Australia
Andrew Gordon, University of Southern California, USA
Pat Hayes, Institute for Human and Machine Cognition, USA
Jerry Hobbs, University of Southern California/ISI, USA
John F. Horty, University of Maryland, USA
David Israel, SRI, USA
Benjamin Johnston, University of Technology, Sydney, Australia
Antonis Kakas, University of Cyprus, Cyprus
Jerome Lang, Centre National de la Recherche Scientifique, France
Joohyung Lee, Arizona State University, USA
Hector Levesque, University of Toronto, Canada
Vladimir Lifschitz, University of Texas at Austin, USA
Sheila McIlraith, University of Toronto, Canada
Rob Miller, University College of London, United Kingdom
Tim Oates, University of Maryland, USA
Pavlos Peppas, University of Patras, Greece
Fiora Pirri, University of Rome, Italy
Erik Sandewall, Linkoping University, Sweden
Sebastian Sardina, RMIT University, Australia
Len Schubert, University of Rochester, USA
Michael Thielscher, Dresden University of Technology, Germany
Rich Thomason, University of Michigan, USA
Achille Varzi, Columbia University, USA

Invited Talks

Acquiring Commonsense Knowledge from Perceptual Observations

Anthony G. Cohen, University of Leeds

Crucial to the ultimate attainment of the goal of building an autonomous cognitive agent is endowing the agent with an ability to perceive, understand, formulate hypotheses and act based on the agent's perceptions. I will discuss work undertaken at Leeds in pursuit of this goal. A key focus of our work is to integrate quantitative and qualitative modes of representation and to learn as much as possible from observation of the world, and thus to acquire high level symbolic models. As one example of our approach, I will show how by characterizing video sequences using a qualitative spatio-temporal relational descriptions, event classes can be mined, and in turn how a taxonomy of functional object categories can be induced from these event descriptions.

Commonsense Reasoning about Chemistry Experiments: Ontology and Representation

Ernest Davis, New York University

How should matter be conceptualized to best support commonsense reasoning about simple physics and chemistry experiments? To address the question, we consider a sheaf of eleven benchmark physical concepts, rules, and scenarios: Part/whole relations among bodies of matter; additivity of mass; motion of a rigid solid object; continuous motion of fluids; fixed mass proportions and spatial continuity at chemical reactions; conservation of mass at chemical reactions; gasses in a container attaining equilibrium; the ideal gas law and the law of partial pressures; liquid at rest in an open container; carrying liquid in an open container; the constant availability of oxygen for reactions in an atmosphere; and surface passivization of metals. We then present a number of different ontologies and representational schemes: the model of atoms and molecules with statistical mechanics; models of spatio-temporal fields, with either points, regions, or histories; models of continuous moving material in terms of chunks of matter, with or without point particles; and a hybrid theory that combines atoms and molecules, chunks of matter, and continuous fields using each where appropriate. We evaluate each of the representational schemes in terms of the ease of representing the benchmark problems and other features. Overall, the field model with histories and hybrid model seem to be best, though neither is unproblematic. We conclude by discussing the major challenges for extending this work.

Diagnosis Revisited

Sheila McIlraith, University of Toronto

In 1987, Ray Reiter proposed a logical characterization of diagnosis from first principles that has had significant influence on the study of diagnostic problem solving. Together with de Kleer and Mackworth he extended this characterization in 1992. Since that time there have been several attempts to build on his fundamental work by creating a characterization of diagnosis of dynamical systems. In this talk, we revisit Reiter's original work on diagnosis, as well as more recent work on diagnosis of dynamical systems. We discuss potential shortcomings of this more recent work, and propose a more general formulation of dynamical diagnosis together with some associated computational machinery.

Solving the Wise Mountain Man Riddle with Answer Set Programming

Marcello Balducini
Intelligent Systems, OCTO
Eastman Kodak Company
Rochester, NY 14650-2102 USA
marcello.balduccini@gmail.com

Abstract

This paper describes an exercise in the formalization of common-sense with Answer Set Programming aimed at solving an interesting riddle, whose solution is not obvious to many people. Solving the riddle requires a considerable amount of common-sense knowledge and sophisticated knowledge representation and reasoning techniques, including planning and adversarial reasoning. Most importantly, the riddle is difficult enough to make it unclear, at a first analysis, whether and how Answer Set Programming or other formalisms can be used to solve it.

Introduction

This paper describes an exercise in the formalization of common-sense with Answer Set Programming (ASP), aimed at solving the riddle:

"A long, long time ago, two cowboys where fighting to marry the daughter of the OK Corral rancher. The rancher, who liked neither of these two men to become his future son-in-law, came up with a clever plan. A horse race would determine who would be allowed his daughter's hand. Both cowboys had to travel from Kansas City to the OK Corral, and the one whose horse arrived LAST would be proclaimed the winner.

The two cowboys, realizing that this could become a very lengthy expedition, finally decided to consult the Wise Mountain Man. They explained to him the situation, upon which the Wise Mountain Man raised his cane and spoke four wise words. Relieved, the two cowboys left his cabin: They were ready for the contest!

Which four wise words did the Wise Mountain Man speak?"

This riddle is interesting because it is easy to understand, but not trivial, and the solution is not obvious to many people. The story can be simplified in various ways without losing the key points. The story is also entirely based on common-sense knowledge. The amount of knowledge that needs to be encoded is not large, which simplifies the encoding; on the other hand, as we will see in the rest of the paper, properly dealing with the riddle requires various sophisticated capabilities, including modeling direct and indirect effects of actions, encoding triggers, planning, dealing with defaults and their exceptions, and concepts from multi-agent systems such as adversarial reasoning. The riddle is difficult enough to make it unclear, at a first analysis, whether and how ASP or other formalisms can be used to formalize the story and underlying reasoning.

In the course of this paper we will discuss how the effects of the actions involved in the story can be formalized, and how to address the main issues of determining that "this could be a lengthy expedition" and of answering the final question.

We begin with a brief introduction on ASP. Next, we show how the knowledge about the riddle is encoded and how reasoning techniques can be used to solve the riddle. Finally, we draw conclusions.

Background

ASP (Marek and Truszczynski 1999) is a programming paradigm based on language A-Prolog (Gelfond and Lifschitz 1991) and its extensions (Balduccini and Gelfond 2003; Brewka, Niemela, and Syrjanen 2004; Mellarkod, Gelfond, and Zhang 2008). In this paper we use the extension of A-Prolog called CR-Prolog (Balduccini and Gelfond 2003), which allows, among other things, simplified handling of exceptions, rare events. To save space, we describe only the fragment of CR-Prolog that will be used in this paper.

Let Σ be a signature containing constant, function and predicate symbols. Terms and atoms are formed as usual. A literal is either an atom a or its strong (also called classical or epistemic) negation $\neg a$.

A *regular rule* (rule, for short) is a statement of the form:

$$h_1 \ \vee \ \ldots \ \vee \ h_k \leftarrow l_1, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n$$

where h_i's and l_i's are literals and *not* is the so-called *default negation*.[1] The intuitive meaning of a rule is that a reasoner, who believes $\{l_1, \ldots, l_m\}$ and has no reason to believe $\{l_{m+1}, \ldots, l_n\}$, has to believe one of h_i's.

A *consistency restoring rule* (cr-rule) is a statement of the form:

$$h_1 \ \vee \ \ldots \ \vee \ h_k \overset{+}{\leftarrow} l_1, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n$$

where h_i's and l_i's are as before. The informal meaning of a cr-rule is that a reasoner, who believes $\{l_1, \ldots, l_m\}$ and

[1] We also allow the use of SMODELS style choice rules, but omit their formal definition to save space.

has no reason to believe $\{l_{m+1}, \ldots, l_n\}$, may believe one of h_i's, but only if strictly necessary, that is only if no consistent set of beliefs can be formed otherwise.

A *program* is a pair $\langle \Sigma, \Pi \rangle$, where Σ is a signature and Π is a set of rules and cr-rules over Σ. Often we denote programs by just the second element of the pair, and let the signature be defined implicitly.

Given a CR-Prolog program Π, we denote the set of its regular rules by Π^r and the set of its cr-rules by Π^{cr}. By $\alpha(r)$ we denote the regular rule obtained from cr-rule r by replacing the symbol $\overset{+}{\leftarrow}$ with $\leftarrow$. Given a set of cr-rules R, $\alpha(R)$ denotes the set obtained by applying α to each cr-rule in R. The semantics of a CR-Prolog program is defined in two steps.

Definition 1 *Given a CR-Prolog program* Π*, a minimal (with respect to set-theoretic inclusion) set* R *of cr-rules of* Π*, such that* $\Pi^r \cup \alpha(R)$ *is consistent is called an* abductive support *of* Π.

Definition 2 *Given a CR-Prolog program* Π*, a set of literals* A *is an* answer set *of* Π *if it is an answer set of the program* $\Pi^r \cup \alpha(R)$ *for some abductive support* R *of* Π.

To represent knowledge and reason about dynamic domains, we use ASP to encode dynamic laws, state constraints and executability conditions (Gelfond and Lifschitz 1998). The laws are written directly in ASP, rather than represented using an action language (Gelfond 2002), to save space and have a more uniform representation.

The key elements of the representation are as follows; we refer the readers to e.g. (Gelfond 2002) for more details. The evolution of a dynamic domain is viewed as a *transition diagram*, which is encoded in a compact way by means of an *action description* consisting of dynamic laws (describing the direct effects of actions), state constraints (describing the indirect effects), and executability conditions (stating when the actions can be executed). Properties of interest, whose truth value changes over time, are represented by *fluents* (e.g. $on(block_1, block_2)$). A state of the transition diagram is encoded as a consistent and complete set of fluent literals (i.e. fluents and their negations). The truth value of a fluent f is encoded by a statement of the form $h(f, s)$, where s is an integer denoting the step in the evolution of the domain, intuitively saying that f holds at step s. The fact that f is false is denoted by $\neg h(f, s)$. Occurrences of actions are traditionally represented by expressions of the form $o(a, s)$, saying that a occurs at step s.

Formalizing the Riddle

The next step is to encode the knowledge about the domain of the story. To focus on the main issues, we abstract from several details and concentrate on the horse ride. The objects of interest are the two competitors (a, b), the two horses ($h(a)$, $h(b)$), and locations $start$, $finish$, and en_route. Horse ownership is described by relation $owns$, defined by the rule $owns(C, h(C)) \leftarrow competitor(C)$.

The fluents of interest and their informal meanings are: $at(X, L)$, "competitor or horse X is at location L"; $riding(C, H)$, "competitor C is riding horse H"; $crossed(X)$, "competitor or horse X has crossed the finish line."

The actions of interest are $wait$, $move$ (the actor moves to the next location along the race track), and $cross$ (the actor crosses the finish line). Because this domain involves multiple actors, we represent the occurrence of actions by a relation $o(A, C, S)$, which intuitively says that action A occurred, performed by competitor C, at step S.[2]

The formalization of action $move$ deserves some discussion. Typically, it is difficult to predict who will complete a race first, as many variables influence the result of a race. To keep our formalization simple, we have chosen a rather coarse-grained model of the movements from one location to the other. Because often one horse will be faster than the other, we introduce a relation $faster(H)$, which informally says that H is the faster horse. This allows us to deal with both simple and more complex situations: when it is known which horse is faster, we encode the information as a fact. When the information is not available, we use the disjunction $faster(h(a)) \ \vee \ faster(h(b))$. Action $move$ is formalized so that, when executed, the slower horse moves from location $start$ to en_route and from en_route to $finish$. The faster horse, instead, moves from $start$ directly to $finish$.[3] The direct effects of the actions can be formalized in ASP as follows:[4]

- Action $move$:

  % If competitor C is at start and riding the faster horse,
  % action $move$ takes him to the finish line.
  $h(at(C, finish), S+1) \leftarrow$
  $\quad h(at(C, start), S),$
  $\quad h(riding(C, H), S),$
  $\quad faster(H),$
  $\quad o(move, C, S).$

  % If competitor C is at start and riding the slower horse,
  % action $move$ takes him to location "en route."
  $h(at(C, en\_route), S+1) \leftarrow$
  $\quad h(at(C, start), S),$
  $\quad h(riding(C, H), S),$
  $\quad$ not $faster(H),$
  $\quad o(move, C, S).$

[2] This simple representation is justified because the domain does not include exogenous actions. Otherwise, we would have to use a more sophisticated representation, such as specifying the actor as an argument of the terms representing the actions.

[3] More refined modeling is possible, but is out of the scope of the present discussion. However, we would like to mention the possibility of using the recent advances in integrating ASP and constraint satisfaction (Mellarkod, Gelfond, and Zhang 2008) to introduce numerical distances, speed, and to take into account parameters such as stamina in their computation.

[4] Depending on the context, executability conditions might be needed stating that each competitor must be riding in order to perform the $move$ or $cross$ actions. Because the story assumes that the competitors are riding at all times, we omit such executability conditions to save space.

% Performing $move$ while "en route" takes the actor
% to the finish line.

$$\begin{array}{l} h(at(C, finish), S+1) \leftarrow \\ \qquad h(at(C, en_{r}oute), S), \\ \qquad o(move, C, S). \end{array}$$

% $move$ cannot be executed while at the finish line.

$$\leftarrow o(move, C, S), h(at(C, finish), S).$$

- Action $cross$:

% Action $cross$, at the finish line, causes the actor to
% cross the finish line.

$$\begin{array}{l} h(crossed(C), S+1) \leftarrow \\ \qquad o(cross, C, S), \\ \qquad h(at(C, finish), S). \end{array}$$

% $cross$ can only be executed at the finish line.

$$\leftarrow o(cross, C, S), h(at(C, L), S), L \neq finish.$$

% $cross$ can be executed only once by each competitor.

$$\leftarrow o(cross, C, S), h(crossed(C), S).$$

No rules are needed for action $wait$, as it has no direct effects. The state constraints are:

- "Each competitor or horse can only be at one location at a time."

$$\begin{array}{l} \neg h(at(X, L_2), S) \leftarrow \\ \qquad h(at(X, L_1), S), \\ \qquad L_1 \neq L_2. \end{array}$$

- "The competitor and the horse he is riding on are always at the same location."

$$\begin{array}{l} h(at(H, L), S) \leftarrow \\ \qquad h(at(C, L), S), \\ \qquad h(riding(C, H), S). \end{array}$$

$$\begin{array}{l} h(at(C, L), S) \leftarrow \\ \qquad h(at(H, L), S), \\ \qquad h(riding(C, H), S). \end{array}$$

It is worth noting that, in this formalization, horses do not perform actions on their own (that is, they are viewed as "vehicles"). Because of that, only the first of the two rules above is really needed. However, the second rule makes the formalization more general, as it allows one to apply it to cases when the horses can autonomously decide to perform actions (e.g. the horse suddenly moves to the next location and the rider is carried there as a side-effect).

- "Each competitor can only ride one horse at a time; each horse can only have one rider at a time."

$$\begin{array}{l} \neg h(riding(X, H2), S) \leftarrow \\ \qquad h(riding(X, H1), S), \\ \qquad H1 \neq H2. \end{array}$$

$$\begin{array}{l} \neg h(riding(C2, H), S) \leftarrow \\ \qquad h(riding(C1, H), S), \\ \qquad C1 \neq C2. \end{array}$$

- "The competitor and the horse he is riding on always cross the finish line together."

$$\begin{array}{l} h(crossed(H), S) \leftarrow \\ \qquad h(crossed(C), S), \\ \qquad h(riding(C, H), S). \end{array}$$

$$\begin{array}{l} h(crossed(C), S) \leftarrow \\ \qquad h(crossed(H), S), \\ \qquad h(riding(C, H), S). \end{array}$$

As noted for the previous group of state constraints, only the first of these two rules is strictly necessary, although the seconds increases the generality of the formalization.

The action description is completed by the law of inertia (Hayes and McCarthy 1969), in its usual ASP representation (e.g. (Gelfond 2002)):

$$h(F, S+1) \leftarrow h(F, S), \text{not } \neg h(F, S+1).$$

$$\neg h(F, S+1) \leftarrow \neg h(F, S), \text{not } h(F, S+1).$$

Reasoning About the Riddle

Let us now see how action description $\mathcal{AD}$, consisting of all of the rules from the previous section, is used to reason about the riddle.

The first task that we want to be able to perform is determining the winner of the race, based on the statement from the riddle "the one whose horse arrived LAST would be proclaimed the winner." In terms of the formalization developed so far, arriving last means being the last to cross the finish line. Encoding the basic idea behind this notion is not difficult, but attention must be paid to the special case of the two horses crossing the finish line together. Commonsense seems to entail that, if the two horses cross the line together, then they are both first. (One way to convince oneself about this is to observe that the other option is to say that both horses arrived last. But talking about "last" appears to imply that they have been preceded by some horse that arrived "first.") The corresponding definition of relations $first\_to\_cross$ and $last\_to\_cross$ is:[5]

% $first\_to\_cross(H)$: horse H crossed the line first.

$$\begin{array}{l} first\_to\_cross(H_1) \leftarrow \\ \qquad h(crossed(H_1), S_2), \\ \qquad \neg h(crossed(H_2), S_1), \\ \qquad S_2 = S_1 + 1, \\ \qquad horse(H_2), H_1 \neq H_2. \end{array}$$

% $last\_to\_cross(H)$: horse H crossed the line last.

$$\begin{array}{l} last\_to\_cross(H_1) \leftarrow \\ \qquad h(crossed(H_1), S_2), \\ \qquad \neg h(crossed(H_1), S_1), \\ \qquad S_2 = S_1 + 1, \\ \qquad h(crossed(H_2), S_1), horse(H_2), H_1 \neq H_2. \end{array}$$

Winners and losers can be determined from the previous relations, and from horse ownership:

% C wins if his horse crosses the finish line last.

$$wins(C) \leftarrow owns(C, H), last\_to\_cross(H).$$

[5] To save space, the definitions of these relations are given for the special case of a 2-competitor race. Extending the definitions to the general case is not difficult, but requires some extra rules.

% C loses if his horse crosses the finish line first.
$loses(C) \leftarrow owns(C, H), first\_to\_cross(H).$

Let $\mathcal{W}$ be the set consisting of the definitions of $last\_to\_cross$, $first\_to\_cross$, $wins$, and $loses$. It is not difficult to check that, given suitable input about the initial state, $\mathcal{AD} \cup \mathcal{W}$ entails intuitively correct conclusions. For example, let σ_0 denote the intended initial state of the riddle, where each competitor is at the start location, riding his horse:

$$h(at(a, start), 0). \quad h(at(b, start), 0).$$

$$\begin{array}{l} h(riding(C, H), 0) \leftarrow \\ \qquad owns(C, H), \\ \qquad \text{not } \neg h(riding(C, H), 0). \end{array}$$

$$\neg h(F, 0) \leftarrow \text{not } h(F, 0).$$

The rule about fluent $riding$ captures the intuition that normally one competitor rides his own horse, but there may be exceptions. Also notice that the last rule in σ encodes the Closed World Assumption, and provides a compact way to specify the fluents that are false in σ. Also, notice that it is not necessary to specify explicitly the location of the horses, as that will be derived from the locations of their riders by state constraints of $\mathcal{AD}$. Assuming that a's horse is the faster, let $F^a = \{faster(h(a))\}$. Let also O^0 denote the set $\{o(a, move, 0), o(b, move, 0)\}$. It is not difficult to see that $\sigma \cup F^a \cup O^0 \cup \mathcal{AD} \cup \mathcal{W}$ entails:

$$\{h(at(a, finish), 1), h(at(b, en\_route), 1)\},$$

meaning that a is expected to arrive at the finish, and b at location "en route." Similarly, given

$$O^1 = \left\{ \begin{array}{ll} o(a, move, 0). & o(b, move, 0). \\ o(a, wait, 1). & o(b, move, 1). \\ o(a, wait, 2). & o(b, cross, 2). \\ o(a, cross, 3). & \end{array} \right.$$

the theory $\sigma \cup F^a \cup O^1 \cup \mathcal{AD} \cup \mathcal{W}$ entails:

$$\begin{array}{l} \{h(at(a, finish), 1),\ h(at(b, finish), 2), \\ \ h(crossed(a), 4),\ h(crossed(b), 3), \\ \ last(h(a)),\ first(h(b)), \\ \ wins(a),\ loses(b)\}, \end{array}$$

meaning that both competitors crossed the finish line, but b's horse crossed it first, and therefore b lost the race.

The next task of interest is to use the theory developed so far to determine that the race "could become a very lengthy expedition." Attention must be paid to the interpretation of this sentence. Intuitively, the sentence refers to the fact that none of the competitors might be able to end the race. However, this makes sense only if interpreted with commonsense. Of course sequences of actions exist that cause the race to terminate. For example, one competitor could ride his horse as fast as he can to the finish line and then cross, but that is likely to cause him to lose the race.

We believe the correct interpretation of the sentence is that we need to check if the two competitors *acting rationally* (i.e. selecting actions in order to achieve their own goal) will ever complete the race. In the remainder of the discussion, we call this the *completion problem*. Notice that, under the assumption of rational acting, no competitor will just run as fast as he can to the finish line and cross it, without paying attention to where the other competitor is.

In this paper, we will focus on addressing the completion problem from the point of view of one of the competitors. That is, we are interested in the reasoning that one competitor needs to perform to solve the problem. So, we will define a relation me, e.g. $me(a)$. In the rest of the discussion, we refer to the competitor whose reasoning we are examining as "our competitor," while the other competitor is referred to as the "adversary."

The action selection performed by our competitor can be formalized using the well-known ASP planning technique (e.g. (Gelfond 2002)) based on a generate-and-test approach, encoded by the set $\mathcal{P}_{me}$ of rules:

$$me(a).$$

$$\begin{array}{l} 1\{\ o(A, C, S) : relevant(A)\ \}1 \leftarrow me(C). \\ \leftarrow \text{not } wins(C), me(C), selected\_goal(win). \end{array}$$

$$relevant(wait).\ relevant(move).\ relevant(cross).$$

where the first rule informally states that the agent should consider performing any action relevant to the task (and exactly one at a time), while the second rule says that sequences of actions that do not lead our competitor to a win should be discarded (if our competitor's goal is indeed to win). Relation $relevant$ allows one to specify which actions are relevant to the task at hand, thus reducing the number of combinations that the reasoner considers.

Our competitor also needs to reason about his adversary's actions. For that purpose, our competitor possesses a model of the adversary's behavior.[6] The model is based on the following heuristics:

- Reach the finish line;
- At the finish line, if the opponent has already crossed, cross (as the race is over anyway);
- At the finish line, if riding the opponent's horse, cross right away;
- Otherwise, wait.

This model of the adversary's behavior could be more sophisticated – for example, it could include some level of non-determinism – but the simple model shown above is sufficient to solve the completion problem for this simple riddle. The heuristics are encoded by the set $\mathcal{P}_{adv}$ of triggers:[7]

$$my\_adversary(C_2) \leftarrow me(C_1), C_1 \neq C_2.$$

$$\begin{array}{l} o(move, C, S) \leftarrow \\ \qquad my\_adversary(C), \\ \qquad \neg h(at(C, finish\_line), S). \end{array}$$

[6] The model here is hard-coded, but could be learned, e.g. (Sakama 2005; Balduccini 2007).

[7] A discussion on the use of triggers can be found in the Conclusions section.

$$\begin{array}{l} o(cross, C_1, S) \leftarrow \\ \quad my\_adversary(C_1), \\ \quad h(at(C_1, finish), S), \\ \quad \neg h(crossed(C_1), S), \\ \quad h(riding(C_1, H), S), \\ \quad owns(C_2, H), C1 \neq C_2. \end{array}$$

$$\begin{array}{l} o(cross, C_1, S) \leftarrow \\ \quad my\_adversary(C_1), \\ \quad h(at(C_1, finish), S), \\ \quad \neg h(crossed(C_1), S), \\ \quad h(crossed(C_2), S), \\ \quad competitor(C_2), C_1 \neq C_2. \end{array}$$

$$\begin{array}{l} \neg o(A_2, C, S) \leftarrow \\ \quad my\_adversary(C), \\ \quad o(A_1, C, S), \\ \quad A_2 \neq A_1. \end{array}$$

$$\begin{array}{l} o(wait, C, S) \leftarrow \\ \quad my\_adversary(C), \\ \quad \text{not } \neg o(wait, C, S). \end{array}$$

Now let us see how the theory developed so far can be used to reason about the completion problem. Let $\mathcal{P}$ denote the set $\mathcal{P}_{me} \cup \mathcal{P}_{adv}$. It is not difficult to see that the theory

$$\sigma \cup F^a \cup \mathcal{AD} \cup \mathcal{W} \cup \mathcal{P}$$

is inconsistent. That is, a has no way of winning if his horse is faster. Let us now show that the result does not depend upon the horse's speed. Let $F^\vee$ denote the rule

$$faster(h(a)) \ \vee \ faster(h(b)).$$

which informally says that it is not known which horse is faster. The theory

$$\sigma \cup F^\vee \cup \mathcal{AD} \cup \mathcal{W} \cup \mathcal{P}$$

is still inconsistent. That is, a cannot win no matter whose horse is faster. Therefore, because our competitor is acting rationally, he is not going to take part in the race. Because the domain of the race is fully symmetrical, it is not difficult to see that b cannot win either, and therefore we will refuse to take part in the race as well.

However, that is not exactly what statement of the completion problem talks about. The statement in fact seems to suggest that, were the competitors to take part in the race (for example, because they hope for a mistake by the opponent), they would not be able to complete the race. To model that, we allow our competitor to have two goals with a preference relation among them: the goal to win, and the goal to at least not lose, where the former is preferred to the second. The second goal formalizes the strategy of waiting for a mistake by the adversary. To introduce the second goal and the preference, we obtain $\mathcal{P}'$ from $\mathcal{P}$ by adding to it the rules:

$$\begin{array}{l} selected\_goal(win) \leftarrow \\ \quad \text{not } \neg selected\_goal(win). \\ \neg selected\_goal(win) \leftarrow \\ \quad selected\_goal(not\_lose). \\ \leftarrow lose(C), me(C), selected\_goal(not\_lose). \\ selected\_goal(not\_lose) \overset{+}{\leftarrow} . \end{array}$$

The first rule says that our competitor's goal is to win, unless otherwise stated. The second rule says that one exception to this is if the selected goal is to not lose. The constraint says that, if the competitor's goal is to not lose, all action selections causing a loss must be discarded. The last rule says that our competitor may possibly decide to select the goal to just not lose, but only if strictly necessary (that is, if the goal of winning cannot be currently achieved).

Now, it can be shown that the theory

$$\sigma \cup F^\vee \cup \mathcal{AD} \cup \mathcal{W} \cup \mathcal{P}'$$

is consistent. One of its answer sets includes for example the atoms:

$$\begin{array}{ll} \{faster(h(a)), & \\ \ o(wait, a, 0), & o(move, b, 0), \\ \ o(wait, a, 1), & o(move, b, 1), \\ \ o(move, a, 2), & o(wait, b, 2), \\ \ o(wait, a, 3), & o(wait, b, 3), \\ \ o(wait, a, 4), & o(wait, b, 4)\ \} \end{array}$$

which represent the possibility that, if a's horse is faster, a and b will reach the finish line, and then wait there indefinitely. To confirm that the race will not be completed, let us introduce a set of rules $\mathcal{C}$ containing the definition of completion, together with a constraint that requires the race to be completed in any model of the underlying theory:

$$\begin{array}{l} completed \leftarrow h(crossed(X), S). \\ \leftarrow \text{not } completed. \end{array}$$

The first rule states that the race has been completed when one competitor has crossed the finish line (the result of the race at that point is fully determined). Because the theory

$$\sigma \cup F^\vee \cup \mathcal{AD} \cup \mathcal{W} \cup \mathcal{P}' \cup \mathcal{C}$$

is inconsistent, we can conclude formally that, if the competitors act rationally, they will not complete the race.

The last problem left to solve is answering the question "Which four wise words did the Wise Mountain Man speak?" In terms of our formalization, we need to find additional information, to be included in the theory developed so far, that allows to entail the completion of the race. Notice that, often, to solve a riddle one needs to revisit assumptions that were initially taken for granted. From a knowledge representation perspective, that means revisiting the defaults used in the encoding of the theory, and allowing the reasoner to select appropriate exceptions to the defaults.

The simple formalization given so far contains only one default, the rule for fluent $riding$ in σ:

$$\begin{array}{l} h(riding(C, H), 0) \leftarrow \\ \quad owns(C, H), \\ \quad \text{not } \neg h(riding(C, H), 0). \end{array}$$

To allow the reasoner to consider exceptions to this default, we add a cr-rule stating that a competitor may possibly ride the opponent's horse, although that should happen only if strictly necessary.

$$\begin{array}{l} h(riding(C, H2), 0) \overset{+}{\leftarrow} \\ \quad owns(C, H1), horse(H2), H1 \neq H2. \end{array}$$

We use a cr-rule, instead of a regular rule, to capture the intuition that the competitors will not normally switch horses. *Although for simplicity here we focus on a specific default, it is important to stress that this technique can be extended to the general case by writing the knowledge base so that each default is accompanied by a cr-rule allowing the reasoner to consider unexpected exceptions (but only if strictly necessary).* Let σ' be obtained from σ by adding to it the new cr-rule. It can be shown that the theory[8]

$$\sigma' \cup F^{\vee} \cup \mathcal{AD} \cup \mathcal{W} \cup \mathcal{P}$$

is consistent and its unique answer set contains:

$$\begin{array}{ll}
\{faster(h(b)), & \\
\ h(riding(a,h(b)),0), & h(riding(b,h(a)),0), \\
\ o(move,a,0), & o(move,b,0), \\
\ o(cross,a,1), & o(move,b,1), \\
\ o(wait,a,2), & o(cross,b,2), \\
\ o(wait,a,3), & o(wait,b,3), \\
\ o(wait,a,4), & o(wait,b,4)\ \}
\end{array}$$

which encodes the answer that, if the competitors switch horses and the horse owned by b is faster, then a can win by immediately reaching the finish line and crossing it. In agreement with common-sense, a does not expect to win if the horse b owns is slower. On the other hand, it is not difficult to see that b will win in that case. That is, the race will be completed no matter what.

The conclusion obtained formally here agrees with the accepted solution of the riddle: "Take each other's horse."

Conclusions

In this paper we have described an exercise in the use of ASP for common-sense knowledge representation and reasoning, aimed at formalizing and reasoning about an easy-to-understand, but non-trivial riddle. One reason why we have selected this particular riddle, besides its high content of common-sense knowledge, is the fact that upon an initial analysis, it was unclear whether and how ASP or other formalisms could be used to solve it. Solving the riddle has required the combined use of some of the latest ASP techniques, including using consistency restoring rules to allow the reasoner to select alternative goals and to consider exceptions to the defaults in the knowledge base as a last resort, and has shown how ASP can be used for adversarial reasoning by employing it to encode a model of the adversary's behavior.

Another possible way of solving the riddle, not shown here for lack of space, consists in introducing a $switch\_horses$ action, made not relevant by default, but with the possibility to use it if no solution can be found otherwise. Such action would be *cooperative*, in the sense that both competitors would have to perform it together. However, as with many actions of this type in a competitive environment, rationally acting competitors are not always expected to agree to perform the action. An interesting continuation of our exercise will consist of an accurate formalization of this solution of the riddle, which we think may yield useful results in the formalization of sophisticated adversarial reasoning.

One last note should be made regarding the use of triggers to model the adversary's behavior. We hope the present paper has shown the usefulness of this technique and the substantial simplicity of implementation using ASP. This technique has limits, however, due to the fact that an a-priori model is not always available. Intuitively, it is possible to use ASP to allow a competitor to "simulate" the opponent's line of reasoning (e.g. by using choice rules). However, an accurate execution of this idea involves solving a number of non-trivial technical issues. We plan to expand on this topic in a future paper.

References

Balduccini, M., and Gelfond, M. 2003. Logic Programs with Consistency-Restoring Rules. In Doherty, P.; McCarthy, J.; and Williams, M.-A., eds., *International Symposium on Logical Formalization of Commonsense Reasoning*, AAAI 2003 Spring Symposium Series, 9–18.

Balduccini, M. 2007. Learning Action Descriptions with A-Prolog: Action Language C. In Amir, E.; Lifschitz, V.; and Miller, R., eds., *Procs of Logical Formalizations of Commonsense Reasoning, 2007 AAAI Spring Symposium.*

Brewka, G.; Niemela, I.; and Syrjanen, T. 2004. Logic Programs wirh Ordered Disjunction. 20(2):335–357.

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing* 365–385.

Gelfond, M., and Lifschitz, V. 1998. Action Languages. *Electronic Transactions on AI* 3(16).

Gelfond, M. 2002. Representing Knowledge in A-Prolog. In Kakas, A. C., and Sadri, F., eds., *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part II*, volume 2408, 413–451. Springer Verlag, Berlin.

Hayes, P. J., and McCarthy, J. 1969. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In Meltzer, B., and Michie, D., eds., *Machine Intelligence 4*. Edinburgh University Press. 463–502.

Marek, V. W., and Truszczynski, M. 1999. *Stable models and an alternative logic programming paradigm.* The Logic Programming Paradigm: a 25-Year Perspective. Springer Verlag, Berlin. 375–398.

Mellarkod, V. S.; Gelfond, M.; and Zhang, Y. 2008. Integrating Answer Set Programming and Constraint Logic Programming. *Annals of Mathematics and Artificial Intelligence*. (to appear).

Sakama, C. 2005. Induction from answer sets in nonmonotonic logic programs. *ACM Transactions on Computational Logic* 6(2):203–231.

[8] The same answer is obtained by replacing $\mathcal{P}$ by $\mathcal{P}'$. However, doing that would require specifying preferences between the cr-rule just added and the cr-rule in $\mathcal{P}'$. To save space, we use $\mathcal{P}$ to answer the final question of the riddle.

A general framework for revising belief bases using qualitative Jeffrey's rule

Salem Benferhat[1] and **Didier Dubois**[2] and **Henri Prade**[2] and **Mary-Anne Williams**[3]
[1] CRIL-CNRS, UMR 8188, Faculté Jean Perrin, Université d'Artois, Rue Jean Souvraz, 62307 Lens France
[2] IRIT - Université Paul Sabatier, 118 route de Narbonne 31062 Toulouse cedex 09 France
[3] Innovation and Enterprise Research Laboratory University of Technology, Sydney NSW 2007 Australia

Abstract

Intelligent agents require methods to revise their epistemic state as they acquire new information. Jeffrey's rule, which extends conditioning to uncertain inputs, is used to revise probabilistic epistemic states when new information is uncertain. This paper analyses the expressive power of two possibilistic counterparts of Jeffrey's rule for modeling belief revision in intelligent agents. We show that this rule can be used to recover most of the existing approaches proposed in knowledge base revision, such as adjustment, natural belief revision, drastic belief revision, revision of an epistemic by another epistemic state. In addition, we also show that that some recent forms of revision, namely reinforcement operators, can also be recovered in our framework.

Introduction

An intelligent agent's information is often uncertain, inconsistent and incomplete. It is then crucially important to define mechanisms to manage it in response to focusing on a specific problem or in response to the acquisition of new, possibly conflicting, information. The term information covers a broad range of entities such as knowledge, perceptions, beliefs, expectations, preferences, or causal relations. It can describe agents' view of the world, itself, its actions and its understanding of changes.

During the past twenty years, many approaches have been proposed to address the problem of belief change from the axiomatic point of view (e.g., (Gärdenfors 1988), (Darwiche & Pearl 1997)), from the semantics point of view (e.g., (Williams 1994), (Boutilier 1993), (Thielscher 2005)) and from the the computational point of view ((Nebel 1994), (Benferhat *et al.* 2002)).

Due to lack of space, this paper only focuses on the semantics of belief revision in the framework of possibility theory. The basic object in possibility theory is a possibility distribution, which is a mapping from the set of classical interpretations to an ordered structure, usually the interval $[0, 1]$. A possibility distribution rank-orders the potential states of the real world according to their level of plausibility, and represents the information available to an agent.

The revision of a possibility distribution can be viewed as a so-called "transmutation" (Makinson 1994) that modifies the ranking of interpretations so as to give priority to the input information. In particular, two forms of possibilistic revision, called possibilistic revision with partial epistemic states, are investigated as the counterparts of Jeffrey's rule of revision in probability theory. These two forms of possibilistic revision consist in modifying a possibility distribution π with a set of weighted, mutually exclusive formulas $\mu = \{\phi_i, a_i\}$, called partial epistemic states, which express that the possibility of ϕ is equal to a_i. These two forms of revision come down to modifying the possibility π such that each formula ϕ_i is accepted with the prescribed degree a_i. The new degrees a_i's may be either a constant determined for example by an expert, or a function defined for instance with respect to the original possibility degree associated with ϕ_i.

This paper first extends the natural properties described in (Benferhat *et al.* 2002) in order to take into account the new form of the input, namely a partial epistemic state. Then we present two definitions of possibilistic revision operators that naturally extend the two forms of conditioning that have been defined in the possibility theory framework. We also compare possibilistic revision with the counterpart of Jeffrey's rule of conditioning. In its second half, the paper shows that most of existing belief revision operators can be recovered by one of the two forms of possibilistic revision with respect to partial epistemic states.

But first in order to establish the new results, we need to restate the necessary background on possibility theory.

Possibilistic representations of epistemic states

Let L be a finite propositional language with formulas ϕ or ψ. $\vDash$ denotes the (semantical) classical consequence relation. Ω is the set of classical interpretations, and $[\phi]$ is the set of classical models of ϕ.

An epistemic agent is a special kind of intelligent agent, one that at any given moment in time is in a specific epistemic state, e.g. it will have a set of current beliefs which are crafted from its background knowledge, conceptual understanding and its (internal and external) perceptions.

We take the traditional interpretation of beliefs and epistemic states and view an *epistemic state* as a set of beliefs where a belief is a relation between an epistemic agent and an object of belief represented as a logical sentence or a proposition.

There are several common representations of epistemic states such as : well ordered partitions of Ω, probabilistic epistemic states, Grove's systems of spheres, Spohn's Ordinal Conditional Functions (OCF), etc. Throughout this paper we use a general representation of a total preorder, namely a possibility distribution π, which is a mapping from Ω to the interval [0,1].

Indeed a possibility distribution can be used for representing any total preorder, and any operator on a total preorder on Ω can be translated into an operator on a possibility distribution to obtain the same outcome. We will identify operators that require the full power of the $[0, 1]$ scale.

Given an interpretation $\omega \in \Omega$, $\pi(\omega)$ represents the degree of compatibility of ω with the available information (or beliefs) about the real world. $\pi(\omega) = 0$ means that the interpretation ω is impossible, and $\pi(\omega) = 1$ means that nothing prevents ω from being the real world. Interpretations ω where $\pi(\omega) = 1$ are considered to be expected (they are not at all surprising). When $\pi(\omega) > \pi(\omega')$, ω is a preferred candidate to ω' for being the real state of the world. The less $\pi(\omega)$ the less plausible ω or the more different it is to the current world. A possibility distribution π is said to be *normal* if $\exists \omega \in \Omega$, such that $\pi(\omega) = 1$, in other words if at least one interpretation is a fully plausible candidate for being the actual world.

Given a possibility distribution π, the possibility degree of formula ϕ is defined as:

$$\Pi_\pi(\phi) = \max\{\pi(\omega) : \omega \in [\phi]\}.$$

It evaluates the extent to which ϕ is consistent with the available information expressed by π. When there is no ambiguity, we simply write $\Pi(\phi)$ instead of $\Pi_\pi(\phi)$. Note that $\Pi(\phi)$ is evaluated under the assumption that the situation where ϕ is true is as normal as can be (since $\Pi(\phi)$ reflects the maximal plausibility of a model of ϕ).

Given a possibility distribution π, the semantic determination of the content of an epistemic state denoted by *content*(π), is obtained by considering all sentences which are more plausible than their negation, namely:

$$\mathit{content}(\pi) = \{\phi : \Pi(\phi) > \Pi(\neg\phi)\}.$$

Namely, *content*(π) is a classical theory whose models are the interpretations having the highest degrees in π. When π is normalized, models of *content*(π) are interpretations which are completely possible, namely $[\mathit{content}(\pi)] = \{\omega : \pi(\omega) = 1\}$. The sentence ϕ belongs to *content*(π) when ϕ holds in all the most normal or plausible situations (hence ϕ is expected, or accepted as being true).

Lastly, given a formula ϕ, two different types of conditioning (Dubois & Prade 1998) have been defined in possibility theory (when $\Pi(\phi) > 0$):

- In an ordinal setting, we have:

$$\begin{aligned} \pi(\omega \mid_m \phi) &= 1 \text{ if } \pi(\omega) = \Pi(\phi) \text{ and } \omega \vDash p \\ &= \pi(\omega) \text{ if } \pi(\omega) < \Pi(\phi) \text{ and } \omega \vdash p \qquad (1) \\ &= 0 \text{ if } \omega \notin [p]. \end{aligned}$$

  This is the definition of *minimum-based conditioning.*

- In a numerical setting, we get:

$$\begin{aligned} \pi(\omega \mid\!. \ \phi) &= \tfrac{\pi(\omega)}{\Pi(\phi)} \text{ if } \omega \vDash p \\ &= 0 \quad \text{otherwise} \end{aligned} \qquad (2)$$

  This is the definition of *product-based conditioning.*

These two definitions of conditioning satisfy an equation of the form

$$\forall \omega, \pi(\omega) = \pi(\omega \mid \phi) \oplus \Pi(\phi)$$

which is similar to Bayesian conditioning, where $\oplus$ is min and the product respectively. The rule based on the product is much closer to genuine Bayesian conditioning than the qualitative conditioning defined from the minimum which is purely based on the comparison of levels; product-based conditioning requires more of the structure of the unit interval. Besides, when $\Pi(\phi) = 0, \pi(\omega \mid_m \phi) = \pi(\omega \mid\!. \ \phi) = 1, \forall \omega$, by convention.

Iterated semantic revision in possibilistic logic

Belief revision results from the effect of accepting a new piece of information called the input information. In this paper, it is assumed that the current epistemic state (represented by a possibility distribution), and the input information, do not play the same role. The input must be incorporated in the epistemic state. In other words, it takes priority over information in the epistemic state. This asymmetry is expressed by the way the belief change problem is stated, namely the new information alters the epistemic state and not conversely. This asymmetry will appear clearly at the level of belief change operations. This situation is different from the one of information fusion from several sources, where no epistemic state dominates *a priori*. In this context, the use of symmetrical rules is natural especially when the sources are equally reliable.

Jeffrey's rule for revising probability distributions

Reasoning in the presence of new observations is a fundamental issue in reasoning with uncertainty and imprecision. In probability theory, there is a natural method for achieving this task using Jeffrey's rule (Jeffrey 1965). This rule is proposed for revising probability distributions based on the probability kinematics principle whose objective is minimizing change. In this method, beliefs are represented as a probability distribution.

Jeffrey's rule (Jeffrey 1965) provides an effective means to revise a probability distribution p to p' given uncertainty bearing on a set of *mutually exclusive* and *exhaustive* events ϕ_i. Note that when speaking of events, ϕ is short for $[\phi]$. The uncertainty is given in the form of pairs (ϕ_i, a_i) with:

$$P'(\phi_i) = a_i. \tag{3}$$

Jeffrey's method relies on the fact that although there is uncertainty about events ϕ_i, conditional probability of any event $\psi \subseteq \Omega$ given any uncertain event ϕ_i remains the same in the original and the revised distributions. Namely,

$$\forall \phi_i, \forall \psi, P(\psi|\phi_i) = P'(\psi|\phi_i). \tag{4}$$

The underlying interpretation of revision implied by the constraint of Equation 4 is that the revised probability distribution p' must not change conditional probability degrees of any event ϕ given uncertain events ϕ_i. In the probabilistic framework, applying Bayes rule then marginalization allows revision of the possibility degree of any event ψ in the following way:

$$P'(\psi) = \sum_{\phi_i} P'(\phi_i) * \frac{P(\psi, \phi_i)}{P(\phi_i)}. \tag{5}$$

The revised probability distribution p' (known as Jeffrey's rule of conditioning) is the unique distribution that satisfies (3) and (4) (see(Chan & Darwiche 2005)).

Two forms of possibilistic revision based on Jeffrey's rule

The possibilistic counterpart of Jeffrey's rule was introduced in (Dubois & Prade 1991) (see also (Dubois & Prade 1997)), without emphasizing the probability kinematics condition (4) however. There are two natural ways to define a possibilistic revision based on Jeffrey's rule, which naturally extend the two forms of conditioning that exist in possibility theory.

Note that most existing works on belief revision (both from semantics and axiomatics perspectives) assume that the input information is either a propositional formula, or an epistemic state (namely a possibility distribution).

Defining a possibilistic revision based on Jeffrey's rule allows us to define a general framework where the input is a compact partition of the set of interpretations. Namely, the input is of the form $\mu = \{(\phi_i, a_i)\ i = 1, m\}$ where the ϕ_i's are pairwise mutually exclusive formulas. The only requirement is that there exists at least one a_j such that $a_j = 1$. In the following, μ will be called a partial epistemic state. It is partial in the sense that letting $\Pi'_{(\pi')}(\phi_i) = a_i$ does not amount to the full specification of π' over the models of ϕ_i.

Let us first discuss some natural properties of the revision of a possibility distribution π and a new input information $\mu = \{(\phi_i, a_i)\ i = 1, m\}$ to a new possibility distribution denoted by $\pi' = \pi(.|\mu)$. Natural properties for π' are:

$\mathbf{A}_1$ **:** π' should be normalized,

$\mathbf{A}_2$ **:** $\forall(\phi_i, a_i) \in \mu, \Pi'(\phi_i) = a_i$.

$\mathbf{A}_3$ **:** $\forall \omega, \omega' \in [\phi_i]$ then: $\pi(\omega) \geq \pi(\omega')$ then $\pi'(\omega) \geq \pi'(\omega')$,

$\mathbf{A}_4$ **:** If for all $\phi_i, \Pi(\phi_i) = a_i$ then $\forall \omega \in [\phi_i] : \pi(\omega) = \pi'(\omega)$,

$\mathbf{A}_5$ **:** If $\pi(\omega) = 0$ then $\pi'(\omega) = 0$.

$\mathbf{A}_1$ means that the new epistemic state is consistent. $\mathbf{A}_2$ confirms that the input (ϕ, a) is interpreted as a constraint which forces π' to satisfy:

$$\Pi'(\phi_i) = a_i.$$

$\mathbf{A}_3$ means that the new possibility distribution should preserve the previous relative order (in the wide sense) between models of each ϕ_i. A stronger version of $\mathbf{A}_3$ can be defined:

$\mathbf{A}'_3$ **:** $\forall \omega, \omega' \in [\phi_i]$ then: $\pi(\omega) > \pi(\omega')$ iff $\pi'(\omega) > \pi'(\omega')$,

$\mathbf{A}'_3$ clearly extends $\mathbf{CR}_1$**,** $\mathbf{CR}_2$ proposed in (Darwiche & Pearl 1997). $\mathbf{A}_4$ means that when all new beliefs ϕ_i are accepted at their prescribed levels a_i then revision does not affect π. $\mathbf{A}_5$ stipulates that impossible worlds remain impossible after revision. Note that there are no further constraints which relate models of different ϕ_i in the new epistemic state.

The previous properties $\mathbf{A}_1$–$\mathbf{A}_5$ do not guarantee a unique definition of conditioning.

$\mathbf{A}_3$ suggests that the possibilistic revision process can be achieved using several parallel changes with a sure input: First, apply a conditioning (using equations 1 and 2) on each ϕ_i and in order to satisfy $\mathbf{A}_2$, the distribution $\pi(\cdot \mid \neg\phi)$ is denormalized so as to satisfy $\Pi'(\phi_i) = a_i$. Therefore, revising with μ can be achieved using the following definition:

$$\forall \phi_i \in \mu, \forall \omega \models \phi_i, \pi(\omega \mid \mu) = a_i \oplus \pi(\omega \mid_{\oplus} \phi_i) \tag{6}$$

where $\oplus$ is either min or the product, depending on whether conditioning is based on the product or the minimum operator. When $\oplus$ = product (resp. min) the possibilistic revision will be simply called product-based (resp. minimum-based) conditioning with partial epistemic states.

The new degree of models of ϕ_i depends on the relative position of the *a priori* possibility degree of ϕ_i, and the prescribed posterior possibility degree of ϕ_i:

- If $\Pi(\phi_i) \geq a_i$ and when $\oplus$ =min, all interpretations that were originally more plausible than a_i, are forced to level a_i, which means that some strict ordering between models of ϕ_i may be lost. Hence $\mathbf{A}'_3$ is clearly not satisfied. When $\oplus$ =product, all plausibility levels are proportionally shifted down (to the level a_i).

- If $\Pi(\phi_i) < a_i$ the best models of ϕ_i are raised to level a_i. Moreover, when $\oplus$ =product, the plausibility levels of other models are proportionally shifted up (to level a_i).

Relationships with Jeffrey's kinematics properties

Another way to define possibilistic revision is to simply apply the counterpart of Jeffrey's rule of conditioning (Jeffrey 1965). Namely, given an initial possibility distribution π and a partial epistemic state $\mu = \{(\phi_i, a_i)\ i = 1, m\}$ we need to find possibility distributions π' that satisfy:

$$\Pi'(\phi_i) = a_i. \tag{7}$$

and:

$$\forall \phi_i, \forall \psi, \Pi(\psi|_{\oplus}\phi_i) = \Pi'(\psi|_{\oplus}\phi_i), \tag{8}$$

where $\oplus$ is either a minimum or a product. When $\oplus$ is the product then we can show that the possibilistic revision given by (6) is the unique possibility distribution that satisfies (7) and (8). However, it is not the case when $\oplus$ is the minimum.

Recovering existing belief revision frameworks

Standard possibilistic conditioning and adjustment

Clearly, possibilistic revision with partial epistemic states generalizes possibilistic conditioning with a propositional formula ϕ. Indeed, applying possibilistic revision given by (6) with a partial epistemic states $\mu = \{(\phi, 1), (\neg\phi, 0)\}$ gives exactly the same results if one applies equation (1) on ϕ when $\oplus = \min$ (resp. (2) for $\oplus$ = product).

Similarly, possibilistic revision with uncertain input, which corresponds to adjustment (see (Benferhat *et al.* 2002)), is a particular case of possibilistic revision with a partial epistemic state, where the input is of the form $\mu = \{(\phi, 1), (\neg\phi, a)\}$.

Natural belief revision

Let $<_{initial}$ be a total pre-order on the set of epistemic states. Let ϕ be a new piece of information. We denote by $<_N$ the result of applying natural belief revision of $<_{initial}$ by ϕ. Natural belief revision of $<_{initial}$ by ϕ proposed in (Boutilier 1993), also hinted by Spohn (Spohn 1988), proceeds to minimal change of $<_{initial}$ by considering the most plausible models of ϕ in $<_{initial}$ to be the most plausible interpretation in $<_N$. More precisely, $<_N$ is defined as follows:

- $\forall \omega \in min(\Omega, <_{initial}), \forall \omega' \in min(\Omega, <_{initial}), \omega =_N \omega'$
- $\forall \omega \in min(\Omega, <_{initial}), \forall \omega' \notin min(\Omega, <_{initial}), \omega <_N \omega'$
- $\forall \omega \notin min(\Omega, <_{initial}), \forall \omega' \notin min(\Omega, <_{initial}), \omega <_N \omega'$ iff $\omega <_{initial} \omega'$.

To recover natural belief revision, first associate with $<_{initial}$ a compatible positive possibility distribution [1] $\pi_{initial}$, defined by :
$\forall \omega, \omega' \in \Omega, \pi_{initial}(\omega) > \pi_{initial}(\omega')$ iff $\omega <_{initial} \omega'$.
Such $\pi_{initial}$ always exists. Then let a be such that $1 > a > max\{\pi(\omega) : \pi(\omega) \neq 1\}$. Then define $\pi_{<_N}(.) = \pi_{input}(.|_m\mu)$ where $\mu = \{(\phi, 1), (\neg\phi, a)\}$, $\pi_{input}(.|_m\mu)$ is the result applying possibilistic revision given by equation (6) with $\oplus$ =min. Then we can show that $\pi_{<_N}$ indeed encodes natural belief revision, namely:

$$\forall \omega, \omega' \in \Omega, \pi_{<_N}(\omega) > \pi_{<_N}(\omega') \text{ iff } \omega <_N \omega'.$$

Drastic belief revision

Papini (Papini 2000), has considered a stronger constraint (also hinted by Spohn (Spohn 1988)) by imposing that each model of ϕ should be strictly preferred to each countermodel of $\neg\phi$, and moreover the relative ordering between models (resp. countermodels) of p should be preserved. More formally, let us denote by $<_D$ be result of applying drastic belief revision of $<_{initial}$ by ϕ. $<_D$ is defined as follows:

- $\forall \omega, \omega' \in [\phi], \omega <_D \omega'$ iff $\omega <_{initial} \omega'$.
- $\forall \omega, \omega' \notin [\phi], \omega <_D \omega'$ iff $\omega <_{initial} \omega'$.
- $\forall \omega \in [\phi], \forall \omega' \notin [\phi], \omega <_D \omega'$.

To recover drastic belief revision, first associate with $<_{initial}$ a compatible positive possibility distribution π_{input}, as defined above. Let $\Delta(\phi) = \min\{\pi(\omega) : \omega \models p\}$, and a such that $a < \Delta(\phi)$.

Then define $\pi_{<_D}(.) = \pi_{input}(.|.\mu)$ where $\mu = \{(\phi, 1), (\neg\phi, a)\}$, $\pi_{input}(.|_m\mu)$ is the result applying possibilistic revision given by equation (6) with $\oplus$ =product. Then we can show that $\pi_{<_D}$ indeed encodes drastic belief revision, namely:

$$\forall \omega, \omega' \in \Omega, \pi_{<_D}(\omega) > \pi_{<_D}(\omega') \text{ iff } \omega <_D \omega'.$$

A revision of epistemic state by epistemic state

In (Benferhat *et al.* 2000) (see also (Nayak 1994)) a revision of an epistemic state, denoted here by $<_{initial}$, by an input in the form of an epistemic state, denoted here by $<_{input}$, is defined. The obtained result is a new epistemic state, denoted by $<_L$ (L for lexicographic ordering), and defined as follows:

- $\forall \omega, \omega' \in \Omega$, if $\omega <_{input} \omega'$ then $\omega <_L \omega'$.

[1] a possibility distribution π is said to be positive if $\forall \omega, \pi(\omega) > 0$.

- $\forall \omega, \omega' \in \Omega$, if $\omega =_{input} \omega'$ then $\omega <_L \omega'$ iff $\omega <_{initial} \omega'$.

Namely, $<_L$ is obtained by refining $<_{input}$ by $<_{initial}$. For our purpose, we denote $\{E_0, ..., E_n\}$ the partition of Ω induced by $<_{input}$. Namely:

- $\forall i, j \in \{0, ..., n\}, E_i \cap E_j = \emptyset$, and $\bigcup_{i=1,...,n} E_i = \Omega$ (namely, E_i's represent a partition of Ω)
- $\forall i \in \{0, ..., n\}, \forall \omega, \omega' \in E_i$, $\omega =_{input} \omega'$,
- $\forall \omega, \omega' \in \Omega, \omega <_{input} \omega'$ iff $\omega \in E_i, \omega' \in E_j$ and $i < j$.

Let $\pi_{initial}$ and π_{input} be two positive possibility distributions associated respectively with $<_{initial}$ and $<_{input}$.

To recover this revision of an epistemic state by an epistemic state, first define $\pi_{<_L}(.) = \pi_{input}(.|.\mu)$ where $\mu = \{(\phi_{E_i}, \epsilon^i) : i = 0, ..., n\}$ is the result applying possibilistic revision given by equation (6) with $\oplus$ =product. ϕ_{E_i} is a propositional formulas that exactly admits E$_i$ as the set of its models. ϵ_i's are infinitesimal (and by convention $\epsilon^0 = 1$).

Then we can show that $\pi_{<_L}$ indeed encodes $<_L$, namely:

$$\forall \omega, \omega' \in \Omega, \pi_{<_L}(\omega) > \pi_{<_L}(\omega') \text{ iff } \omega <_L \omega'.$$

Reinforcement operator

The last approach that we propose to recover is called a reinforcement operator, recently proposed in (Konieczny & Perez 2008). The idea is that a revision of $<_{initial}$ by a propositional formula ϕ only allows the improvement in plausibility of ϕ, namely the result makes ϕ "one unit" more plausible.

Let new epistemic state , denoted by $<_R$, obtained after reinforcing ϕ is defined as follows:

- The relative orderings between models (resp. counter-models) of ϕ is preserved.
- Let ω be a model of ϕ and ω' be a counter-model of ϕ. :
  - if $\omega' = \omega$ then $\omega <_R \omega'$
  - if $\omega' <_{initial} \omega$ then if $\forall \omega$" $\in [\phi]$ where $\omega' <_{initial}$ ω" we have $\omega \leq \omega$" then $\omega =_R \omega'$ otherwise $\omega' <_R \omega$
  - if $\omega <_{initial} \omega'$ then $\omega <_R \omega'$

To recover the reinforcement operator, we first define π_{input} to be a positive possibility distribution associated with $<_{initial}$, as defined above. Let $S = \{a_0 = 1, a_1, ..., a_n\}$ be a finite scale $1 > a_1 > ... > a_n > 0$ (n is at least equal to twice the number of different degrees in π_{input}. Define pred$(a_i) = a_{i-1}$ with by convention $pred(a_0) = 1$, and succ$(a_i) = a_{i+1}$ with by convention $Succ(a_n) = a_n$.

Let a$_i \in S$ be such that $a_i = \pi_{input}(\phi)$.

Define $\pi_{<_R}(.) = \pi_{initial}(.|.\mu)(.)$ where $\mu = \{(\phi, min(1, pred(\Pi(\phi))), (\neg\phi, succ(\Pi(\neg\phi)) \ominus \Pi(\phi)\}$ is the result of applying possibilistic revision given by equation (6) with $\oplus$ =product, and where "succ$(\Pi(\neg\phi))\ominus\Pi(\phi)$" is defined as equal to 1 if $\Pi(\neg\phi) > \Pi(\phi)$ and equal to $succ(\Pi(\neg\phi))$ otherwise.

Then we can show that $\pi_{<_R}$ indeed encodes $<_R$, namely:

$$\forall \omega, \omega' \in \Omega, \pi_{<_R}(\omega) > \pi_{<_R}(\omega') \text{ iff } \omega <_R \omega'.$$

Conclusion

The information held by an intelligent agent is typically uncertain, inconsistent and incomplete, consequently agents need sophisticated mechanisms to revise their epistemic states as they acquire new information over time because this new information may be in conflict with information in it epistemic state.

Due to the fundamental nature of the need to maintain an epistemic state that faithfully reflects an agents understanding there as been considerable scientific effort invested in developing effective belief revision mechanisms and strategies such as (Gärdenfors 1988), (Darwiche & Pearl 1997), (Williams 1994), (Nebel 1994), and (Benferhat *et al.* 2002).

In this paper we show how Jeffrey's rule can be used to justify several key existing approaches to belief revision, then having established this sound relationship we show that reinforcement operators can be specified using our framework. Lastly, we propose a new form of belief revision where the input is only a partial representation of epistemic states using Jeffrey's rule. All these methods can be used to enhance the belief management capabilities of intelligent agents.

References

Benferhat, S.; Konieczny, S.; Papini, O.; and Pérez, R. P. 2000. Iterated revision by epistemic states: axioms, semantics and syntax. In *Proc. of the 14th European Conf. on Artificial Intelligence (ECAI-00)*, 13–17. Berlin, Allemagne: IOS Press.

Benferhat, S.; Dubois, D.; Prade, H.; and Williams, M.-A. 2002. A practical approach to revising prioritized knowledge bases. *Studia Logica Journal* 70:105–130.

Boutilier, C. 1993. Revision Sequences and Nested Conditionals. In *Proc. of the 13th Inter. Joint Conf. on Artificial Intelligence (IJCAI'93)*, 519–531.

Chan, H., and Darwiche, A. 2005. On the revision of probabilistic beliefs using uncertain evidence. *Artificial Intelligence* 163:67–90.

Darwiche, A., and Pearl, J. 1997. On the logic of iterated revision. *Artificial Intelligence* 89:1–29.

Dubois, D., and Prade, H. 1991. *Updating with belief functions, ordinal conditional functions and possibility measures.* Uncertainty in Artificial Intelligence 6 (P. P. Bonissone, M. Henrion, L. N. Kanal, J. F. Lemmer, eds). Elsevier Science Publ. B.V. 311–329.

Dubois, D., and Prade, H. 1997. A synthetic view of belief revision with uncertain inputs in the framework of possibility theory. *Int. J. Approx. Reasoning* 17:295–324.

Dubois, D., and Prade, H. 1998. Possibility theory: qualitative and quantitative aspects. *Handbook of Defeasible Reasoning and Uncertainty Management Systems. (D. Gabbay, Ph. Smets, eds.), Vol. 1: Quantified Representation of Uncertainty and Imprecision (Ph. Smets, ed.)* 169–226.

Gärdenfors, P. 1988. *Knowledge in Flux: Modeling the Dynamics of Epistemic States.* Bradford Books. Cambridge: MIT Press.

Jeffrey, R. C.. 1965. *The logic of decision.* Mc. Graw Hill, New York.

Konieczny, S., and Perez, R. P. 2008. Improvement operators. In *11th International Conference on Principles of Knowledge Representation and Reasoning(KR'08)*, 177–186.

Makinson, D. 1994. General patterns in nonmonotonic inference. In *Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 3 (D.M. Gabbay, C.J. Hogger, J.A. Robinson, D. Nute, eds.)*, 35–110. Oxford University Press,.

Nayak, A. 1994. Iterated belief change based on epistemic entrenchment. *Erkenntnis* 41:353–390.

Nebel, B. 1994. Base revision operations and schemes: semantics, representation, and complexity. In *Proceedings of the Eleventh European Conference on Artificial Intelligence (ECAI'94)*, 341–345.

Papini, O. 2000. Iterated revision operations stemming from the history of an agent's observations. *Frontiers of Belief Revision* to appear.

Spohn, W. 1988. Ordinal conditiona functions: a dynamic theory of epistemic states. *Causation in Decision, Belief Change, and Statistics* 2:105–134.

Thielscher, M. 2005. Handling implicational and universal quantification constraints in flux. In van Beek., ed., *Proceedings of the International Conference on Principle and Practice of Constraint Programming (CP)*, volume 3709 of *LNCS*, 667–681. Sitges, Spain: Springer.

Williams, M. A. 1994. Transmutations of Knowledge Systems. In Doyle, J., and al. Eds., eds., *Inter. Conf. on principles of Knowledge Representation and reasoning (KR'94)*, 619–629. Morgan Kaufmann.

Next Steps in Propositional Horn Contraction*

Richard Booth
Mahasarakham University
Thailand
richard.b@msu.ac.th

Thomas Meyer
Meraka Institute, CSIR and
School of Computer Science
University of Kwazulu-Natal
South Africa
tommie.meyer@meraka.org.za

Ivan José Varzinczak
Meraka Institute, CSIR
South Africa
ivan.varzinczak@meraka.org.za

Abstract

Standard belief contraction assumes an underlying logic containing full classical propositional logic, but there are good reasons for considering contraction in less expressive logics. In this paper we focus on *Horn logic*. In addition to being of interest in its own right, our choice is motivated by the use of Horn logic in several areas, including ontology reasoning in description logics. We consider three versions of contraction: *entailment-based* and *inconsistency-based* contraction (e-contraction and i-contraction, resp.), introduced by Delgrande for Horn logic, and *package contraction* (p-contraction), studied by Fuhrmann and Hansson for the classical case. We show that the standard basic form of contraction, *partial meet*, is too strong in the Horn case. We define more appropriate notions of basic contraction for all three types above, and provide associated representation results in terms of postulates. Our results stand in contrast to Delgrande's conjectures that *orderly maxichoice* is *the* appropriate contraction for both e- and i-contraction. Our interest in p-contraction stems from its relationship with an important reasoning task in ontological reasoning: *repairing the subsumption hierarchy* in $\mathcal{EL}$. This is closely related to p-contraction with sets of *basic* Horn clauses (Horn clauses of the form $p \rightarrow q$). We show that this restricted version of p-contraction can also be represented as i-contraction.

Introduction

Belief change is a subarea of knowledge representation concerned with describing how an intelligent agent ought to change its beliefs about the world in the face of new and possibly conflicting information. Arguably the most influential work in this area is the so-called AGM approach (Alchourrón, Gärdenfors, and Makinson 1985; Gärdenfors 1988) which focuses on two types of belief change: *belief revision*, in which an agent has to keep its set of beliefs consistent while incorporating new information into it, and *belief contraction*, in which an agent has to give up some of its beliefs in order to avoid drawing unwanted conclusions.

Although belief change is relevant to a wide variety of application areas, most approaches, including AGM, assume an underlying logic which includes full propositional logic. In this paper we deviate from this trend and investigate belief contraction for propositional Horn logic. As pointed out by Delgrande (2008) who has also studied contraction for Horn logic recently, and to whom we shall frequently refer in this paper, this is an important topic for a number of reasons: (i) it sheds light on the theoretical underpinnings of belief change, and (ii) Horn logic has found extensive use in AI and database theory. However, our primary reason for focusing on this topic is because of its application to ontologies in description logics (DLs) (Baader et al. 2003). Horn clauses correspond closely to subsumption statements in DLs (roughly speaking, statements of the form $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B$ where the A_i's and B are *concepts*), especially in the $\mathcal{EL}$ family of DLs (Baader 2003), since both Horn logic and the $\mathcal{EL}$ family lack full negation and disjunction. A typical scenario involves an ontology engineer teaming up with an expert to construct an ontology related to the domain of expertise of the latter with the aid of an ontology engineering tool such as SWOOP [http://code.google.com/p/swoop] or Protégé [http://protege.stanford.edu]. One of the principal methods for testing the quality of a constructed ontology is for the domain expert to inspect and verify the computed *subsumption hierarchy*. Correcting such errors involves the expert pointing out that certain subsumptions are missing from the subsumption hierarchy, while others currently occurring in the subsumption hierarchy ought not to be there. A concrete example of this involves the medical ontology SNOMED (Spackman, Campbell, and Cote 1997) which classifies the concept Amputation-of-Finger as being subsumed by the concept Amputation-of-Arm. Finding a solution to problems such as these is known as *repair* in the DL community (Schlobach and Cornet 2003), but it can also be seen as the problem of *contracting* by the subsumption statement Amputation-of-Finger $\sqsubseteq$ Amputation-of-Arm.

The scenario also illustrates why we are concerned with belief contraction of belief *sets* (logically closed theories) and not *belief base* contraction (Hansson 1999). Ontologies are not constructed by writing DL axioms, but rather using ontology editing tools, from which the axioms are generated automatically. Because of this, it is the belief set and not the axioms from which the theory is generated that is important.

*This work has been accepted for presentation at IJCAI-09. The paper is based upon work supported by the National Research Foundation under Grant number 65152.

Logical Background and Belief Contraction

We work in a finitely generated propositional language $\mathcal{L}_{\mathsf{P}}$ over a set of propositional atoms $\mathfrak{P}$, which includes the distinguished atoms $\top$ and $\bot$, and with the standard model-theoretic semantics. Atoms will be denoted by $p, q, \ldots$, possibly with subscripts. We use $\varphi, \psi, \ldots$ to denote classical formulas. They are recursively defined in the usual way.

We denote by $\mathcal{V}$ the set of all valuations or interpretations $v : \mathfrak{P} \longrightarrow \{0, 1\}$, with 0 denoting falsity and 1 truth. Satisfaction of φ by v is denoted by $v \Vdash \varphi$. The set of models of a set of formulas X is $[X]$. We sometimes represent the valuations of the logic under consideration as sequences of 0s and 1s, and with the obvious implicit ordering of atoms. Thus, for the logic generated from p and q, the valuation in which p is true and q is false will be represented as 10.

Classical logical consequence and logical equivalence are denoted by $\models$ and $\equiv$ respectively. For sets of sentences X and Φ, we understand $X \models \Phi$ to mean that X entails *every* element of Φ. For $X \subseteq \mathcal{L}_{\mathsf{P}}$, the set of sentences logically entailed by X is denoted by $Cn(X)$. A *belief set* is a logically closed set, i.e., for a belief set X, $X = Cn(X)$. $\mathscr{P}(X)$ denotes the power set (set of all subsets) of X.

A *Horn clause* is a sentence of the form $p_1 \wedge p_2 \wedge \ldots \wedge p_n \rightarrow q$ where $n \geq 0$. If $n = 0$ we write q instead of $\rightarrow q$. A *Horn theory* is a set of Horn clauses. Given a propositional language $\mathcal{L}_{\mathsf{P}}$, the Horn language $\mathcal{L}_{\mathsf{H}}$ generated from $\mathcal{L}_{\mathsf{P}}$ is simply the Horn clauses occurring in $\mathcal{L}_{\mathsf{P}}$. The Horn logic obtained from $\mathcal{L}_{\mathsf{H}}$ has the same semantics as the propositional logic obtained from $\mathcal{L}_{\mathsf{P}}$, but just restricted to Horn clauses. Thus a *Horn belief set* is a Horn theory closed under logical entailment, but containing only Horn clauses. Hence, $\models$, $\equiv$, the $Cn(.)$ operator, and all other related notions are defined relative to the logic we are working in (e.g. $\models_{\mathsf{PL}}$ for propositional logic and $\models_{\mathsf{HL}}$ for Horn logic). Since the context always makes it clear which logic we are dealing with, we shall dispense with such subscripts for the sake of readability.

AGM (Alchourrón, Gärdenfors, and Makinson 1985) is the best-known approach to contraction. It gives a set of postulates characterising all rational contraction functions. The aim is to describe belief contraction on the *knowledge level* independent of how beliefs are represented. Belief states are modelled by *belief sets* in a logic with a Tarskian consequence relation including classical propositional logic. The *expansion* of K by φ, $K + \varphi$, is defined as $Cn(K \cup \{\varphi\})$. *Contraction* is intended to represent situations in which an agent has to give up information from its current beliefs. Formally, belief contraction is a (partial) function from $\mathscr{P}(\mathcal{L}_{\mathsf{P}}) \times \mathcal{L}_{\mathsf{P}}$ to $\mathscr{P}(\mathcal{L}_{\mathsf{P}})$: the contraction of a belief set by a sentence yields a new set.

The AGM approach to contraction requires that the following set of postulates characterise *basic* contraction.

$(K{-}1)$ $K - \varphi = Cn(K - \varphi)$

$(K{-}2)$ $K - \varphi \subseteq K$

$(K{-}3)$ If $\varphi \notin K$, then $K - \varphi = K$

$(K{-}4)$ If $\not\models \varphi$, then $\varphi \notin K - \varphi$

$(K{-}5)$ If $\varphi \equiv \psi$, then $K - \varphi = K - \psi$

$(K{-}6)$ If $\varphi \in K$, then $(K - \varphi) + \varphi = K$

The intuitions behind these postulates have been debated in numerous works (Gärdenfors 1988; Hansson 1999). We will not do so here, and just note that $(K{-}6)$, a.k.a. *Recovery*, is the most controversial. There is also a refined version of AGM contraction involving two *extended postulates*, but a discussion on that is beyond the scope of this paper.

Various methods exist for constructing basic AGM contraction. In this paper we focus on the use of *remainder sets*.

Definition 1 *For a belief set K, $X \in K \downarrow \varphi$ iff (i) $X \subseteq K$, (ii) $X \not\models \varphi$, and (iii) for every X' s.t. $X \subset X' \subseteq K$, $X' \models \varphi$. We call the elements of $K \downarrow \varphi$* remainder sets *of K w.r.t. φ.*

It is easy to verify that remainder sets are belief sets, and that remainder sets can be generated semantically by adding precisely one countermodel of φ to the models of K (when such countermodels exist). Also, $K \downarrow \varphi = \emptyset$ iff $\models \varphi$.

Since there is no unique method for choosing between possibly different remainder sets, AGM contraction presupposes the existence of a suitable selection function for doing so.

Definition 2 *A selection function σ is a function from $\mathscr{P}(\mathscr{P}(\mathcal{L}_{\mathsf{P}}))$ to $\mathscr{P}(\mathscr{P}(\mathcal{L}_{\mathsf{P}}))$ s.t. $\sigma(K \downarrow \varphi) = \{K\}$, if $K \downarrow \varphi = \emptyset$, and $\emptyset \neq \sigma(K \downarrow \varphi) \subseteq K \downarrow \varphi$ otherwise.*

Selection functions provide a mechanism for identifying the remainder sets judged to be most appropriate, and the resulting contraction is then obtained by taking the intersection of the chosen remainder sets.

Definition 3 *For σ a selection function, $-_\sigma$ is a partial meet contraction iff $K -_\sigma \varphi = \bigcap \sigma(K \downarrow \varphi)$.*

One of the fundamental results of AGM contraction is a representation theorem which shows that partial meet contraction corresponds exactly with the six basic AGM postulates.

Theorem 1 ((Gärdenfors 1988)) *Every partial meet contraction satisfies $(K{-}1)$–$(K{-}6)$. Conversely, every contraction function satisfying $(K{-}1)$–$(K{-}6)$ is a partial meet contraction.*

Two subclasses of partial meet deserve special mention.

Definition 4 *Given a selection function σ, $-_\sigma$ is a* maxichoice *contraction iff $\sigma(K \downarrow \varphi)$ is a singleton set. It is a* full meet *contraction iff $\sigma(K \downarrow \varphi) = K \downarrow \varphi$ whenever $K \downarrow \varphi \neq \emptyset$.*

Clearly full meet contraction is unique, while maxichoice contraction usually is not. Observe also that partial meet contraction satisfies the following convexity principle.

Proposition 1 *Let K be a belief set, let $-_{mc}$ be a maxichoice contraction, and let $-_{fm}$ be full meet contraction. For every belief set X s.t. $(K -_{fm} \varphi) \subseteq X \subseteq K -_{mc} \varphi$, there is a partial meet contraction $-_{pm}$ such that $K -_{pm} \varphi = X$.*

That is, every belief set between the results from full meet contraction and some maxichoice contraction is obtained from some partial meet contraction. This result plays an important part in our definition of Horn contraction.

Horn Contraction Horn contraction differs from classical AGM contraction in a number of ways. The most basic differences are the use of Horn logic as the underlying logic and allowing for the contraction of finite *sets* of sentences Φ.

As recognised by Delgrande (2008), the move to Horn logic admits the possibility of more than one type of contraction. He considers two types: entailment-based contraction (or e-contraction) and inconsistency-based contraction (or i-contraction). In what follows, we recall Delgrande's approach and develop our theory of Horn contraction.

Entailment-based contraction

For e-contraction, the goal of contracting with a set of sentences Φ is the removal of at least one of the sentences in Φ. For full propositional logic, contraction with a set of sentences is not particularly interesting since contracting by Φ will be equivalent to contracting by the single sentence $\bigwedge \Phi$. For Horn logic it is interesting though, since the conjunction of the sentences in Φ is not always expressible as a single sentence. (An alternative, and equivalent approach, would have been to allow for the *conjunction* of Horn clauses as Delgrande (2008) does, but for reasons that will become clear later, we have not opted for this choice.) Our starting point for defining Horn e-contraction is in terms of Delgrande's definition of e-remainder sets.

Definition 5 (Horn e-Remainder Sets) *For a belief set H, $X \in H \downarrow_e \Phi$ iff (i) $X \subseteq H$, (ii) $X \not\models \Phi$, and (iii) for every X' s.t. $X \subset X' \subseteq H$, $X' \models \Phi$. We refer to the elements of $H \downarrow_e \Phi$ as the* Horn e-remainder sets *of H w.r.t. Φ.*

It is easy to verify that all Horn e-remainder sets are belief sets. Also, $H \downarrow_e \Phi = \emptyset$ iff $\models \Phi$.

We now proceed to define selection functions to be used for Horn partial meet e-contraction.

Definition 6 (Horn e-Selection Functions) *A partial meet Horn e-selection function σ is a function from $\mathscr{P}(\mathscr{P}(\mathcal{L}_\mathsf{H}))$ to $\mathscr{P}(\mathscr{P}(\mathcal{L}_\mathsf{H}))$ s.t. $\sigma(H \downarrow_e \Phi) = \{H\}$ if $H \downarrow_e \Phi = \emptyset$, and $\emptyset \neq \sigma(H \downarrow_e \Phi) \subseteq H \downarrow_e \Phi$ otherwise.*

Using these selection functions, we define partial meet Horn e-contraction.

Definition 7 (Partial Meet Horn e-Contraction) *Given a partial meet Horn e-selection function σ, $-_\sigma$ is a partial meet Horn e-contraction iff $H -_\sigma \Phi = \bigcap \sigma(H \downarrow_e \Phi)$.*

We also consider two special cases.

Definition 8 (Maxichoice and Full Meet) *Given a partial meet Horn e-selection function σ, $-_\sigma$ is a* maxichoice Horn e-contraction *iff $\sigma(H \downarrow_e \Phi)$ is a singleton set. It is a* full meet Horn e-contraction *iff $\sigma(H \downarrow_e \Phi) = H \downarrow_e \Phi$ when $H \downarrow_e \Phi \neq \emptyset$.*

Example 1 *Let $H = Cn(\{p \rightarrow q, q \rightarrow r\})$. Then $H \downarrow_e \{p \rightarrow r\} = \{H', H''\}$, where $H' = Cn(\{p \rightarrow q\})$, and $H'' = Cn(\{q \rightarrow r, p \wedge r \rightarrow q\})$. So contracting with $\{p \rightarrow r\}$ yields either H', H'', or $H' \cap H'' = Cn(\{p \wedge r \rightarrow q\})$.*

Beyond Partial Meet Contraction

While all partial meet e-contractions (and therefore also maxichoice and full meet e-contractions) are appropriate choices for e-contraction, they do not make up the set of *all* appropriate Horn e-contractions. This has a number of implications, one of them being that it conflicts with Delgrande's conjecture that *orderly* maxichoice e-contraction is the appropriate form of e-contraction.

The argument that maxichoice e-contraction is not sufficient is a relatively straightforward one. In full propositional logic the argument against maxichoice contraction relates to the link between AGM *revision* and contraction via the Levi Identity (Levi 1977): $K \star \varphi = (K - \neg\varphi) + \varphi$. For maxichoice contraction this has the unfortunate consequence that a revision operator obtained via the Levi Identity will satisfy the following "fullness result", i.e., $K \star \varphi$ is a *complete* theory: If $\neg\varphi \in K$ then for all $\psi \in \mathcal{L}_\mathsf{P}$, $\psi \in K \star \varphi$ or $\neg\psi \in K \star \varphi$. Semantically, this occurs because the models of any remainder set for φ are obtained by adding a single countermodel of $\neg\varphi$ to the models of K. And while it is true that e-remainder sets for Horn logic do not always have this property, the fact is that they still frequently do, which means that maxichoice e-contraction will frequently cause the same problems as in propositional logic. For example, consider the Horn belief set $H = Cn(\{p, q\})$. It is easy to verify that $[H] = \{11\}$, that the e-remainder sets of $\{p\}$ w.r.t. H are $H' = Cn(\{p \rightarrow q, q \rightarrow p\})$ and $H'' = Cn(\{q\})$, and that $[H'] = \{11, 00\}$ and $[H''] = \{11, 01\}$: i.e., the models of H' and H'' are obtained by adding to the models of H a single countermodel of p. This is not to say that maxichoice e-contraction is *never* appropriate. Similar to the case for full propositional logic, we argue that all maxichoice Horn e-contractions ought to be seen as rational ways of contracting. It is just that other possibilities may be more applicable in certain situations. And, just as in the case for full propositional logic, this leads to the conclusion that all partial meet e-contractions ought to be seen as appropriate.

Once partial meet e-contraction has been accepted as necessary for Horn e-contraction, the obvious next question is whether partial meet Horn e-contraction is sufficient, i.e., whether there are any rational e-contractions that are not partial meet Horn e-contractions. For full propositional logic the sufficiency of partial meet contraction can be justified by Proposition 1 which, as we have seen, states that every belief set between full meet contraction and some maxichoice contraction is obtained from some partial meet contraction. It turns out that the same result does not hold for Horn logic.

Example 2 *As we have seen in Example 1, for the e-contraction of $\{p \rightarrow r\}$ from the Horn belief set $Cn(\{p \rightarrow q, q \rightarrow r\})$, full meet yields $H_{fm} = Cn(\{p \wedge r \rightarrow q\})$ while maxichoice yields either $H^1_{mc} = Cn(\{p \rightarrow q\})$ or $H^2_{mc} = Cn(\{q \rightarrow r, p \wedge r \rightarrow q\})$. Now consider the belief set $H' = Cn(\{p \wedge q \rightarrow r, p \wedge r \rightarrow q\})$. It is clear that $H_{fm} \subseteq H' \subseteq H^2_{mc}$, but there is no partial meet e-contraction yielding H'.*

Our contention is that Horn e-contraction should be extended to include cases such as H' above. Since full meet Horn e-contraction is deemed to be appropriate, it stands to reason that any belief set H' bigger than it should also be seen as appropriate, *provided* that H' does not contain any irrelevant additions. But since H' is contained in some maxichoice Horn e-contraction, H' cannot contain any irrelevant additions. After all, the maxichoice Horn e-contraction contains only relevant additions, since it is an appropriate form of contraction. Hence H' is also an appropriate result of e-contraction.

Definition 9 (Infra e-Remainder Sets) *For belief sets H and X, $X \in H \Downarrow_e \Phi$ iff there is some $X' \in H \downarrow_e \Phi$ s.t. $(\bigcap H \downarrow_e \Phi) \subseteq X \subseteq X'$. We refer to the elements of $H \Downarrow_e \Phi$ as the* infra e-remainder sets *of H w.r.t. Φ.*

Note that all e-remainder sets are also infra e-remainder sets, and so is the intersection of any set of e-remainder sets. Indeed, the intersection of any set of infra e-remainder sets is also an infra e-remainder set. So the set of infra e-remainder sets contains *all* belief sets between some Horn e-remainder set and the intersection of all Horn e-remainder sets. This explains why Horn e-contraction is not defined as the intersection of infra e-remainder sets (cf. Definition 7).

Definition 10 (Horn e-Contraction) *An infra e-selection function τ is a function from $\mathscr{P}(\mathscr{P}(\mathcal{L}_\mathsf{H}))$ to $\mathscr{P}(\mathcal{L}_\mathsf{H})$ s.t. $\tau(H \Downarrow_e \Phi) = H$ whenever $\models \Phi$, and $\tau(H \Downarrow_e \Phi) \in H \Downarrow_e \Phi$ otherwise. A contraction function $-_\tau$ is a Horn e-contraction iff $H -_\tau \Phi = \tau(H \Downarrow_e \Phi)$.*

A Representation Result

Our representation result makes use of all of the basic AGM postulates, except for the Recovery Postulate $(K - 6)$. It is easy to see that Horn e-contraction does not satisfy Recovery. As an example, take $H = Cn(\{p \rightarrow r\})$ and let $\Phi = \{p \wedge q \rightarrow r\}$. Then $H - \Phi = Cn(\emptyset)$ and so $(H -_e \Phi) + \Phi = Cn(\{p \wedge q \rightarrow r\}) \neq H$. In place of Recovery we have a postulate that closely resembles Hansson's (1999) Relevance Postulate, and a postulate handling the case when trying to contract with a tautology.

$(H -_e 1)$ $H -_e \Phi = Cn(H -_e \Phi)$

$(H -_e 2)$ $H -_e \Phi \subseteq H$

$(H -_e 3)$ If $\Phi \not\subseteq H$ then $H -_e \Phi = H$

$(H -_e 4)$ If $\not\models \Phi$ then $\Phi \not\subseteq H -_e \Phi$

$(H -_e 5)$ If $Cn(\Phi) = Cn(\Psi)$ then $H -_e \Phi = H -_e \Psi$

$(H -_e 6)$ If $\varphi \in H \setminus (H -_e \Phi)$ then there is a H' such that $\bigcap(H \downarrow_e \Phi) \subseteq H' \subseteq H$, $H' \not\models \Phi$, and $H' + \{\varphi\} \models \Phi$

$(H -_e 7)$ If $\models \Phi$ then $H -_e \Phi = H$

Postulates $(H -_e 1)$–$(H -_e 5)$ are analogues of $(K{-}1)$–$(K{-}5)$, while $(H -_e 6)$ states that all sentences removed from H during a Φ-contraction must have been removed for a reason: adding them again brings back Φ. $(H -_e 7)$ simply states that contracting with a (possibly empty) set of tautologies leaves the initial belief set unchanged. We remark that $(H -_e 3)$ is actually redundant in the list, since it can be shown to follow mainly from $(H -_e 6)$.

Theorem 2 *Every Horn e-contraction satisfies $(H -_e 1)$–$(H -_e 7)$. Conversely, every contraction function satisfying $(H -_e 1)$–$(H -_e 7)$ is a Horn e-contraction.*

Inconsistency-based Contraction

We now turn our attention to Delgrande's second type of contraction for Horn logic: inconsistency-based contraction, or i-contraction. The purpose of this type of contraction by a set Φ is to modify the belief set H in such a way that adding Φ to H does not result in an inconsistent belief set: $(H -_i \Phi) + \Phi \not\models \bot$. Our starting point for defining i-contraction is in terms of Delgrande's definition of i-remainder sets with respect to Horn logic.

Definition 11 (Horn i-Remainder Sets) *For a belief set H, $X \in H \downarrow_i \Phi$ iff (i) $X \subseteq H$, (ii) $X + \Phi \not\models \bot$, and (iii) for every X' s.t. $X \subset X' \subseteq H$, $X' + \Phi \models \bot$. We refer to the elements of $H \downarrow_i \Phi$ as the* Horn i-remainder sets *of H w.r.t. Φ.*

It is again easy to verify that Horn i-remainder sets are belief sets and that $H \downarrow_i \Phi = \emptyset$ iff $\Phi \models \bot$.

The definition of i-remainder sets is similar enough to that of e-remainder sets (Definition 5) that we can define partial meet Horn i-selection functions, partial meet Horn i-contraction, maxichoice Horn i-contraction, and full meet Horn i-contraction by repeating Definitions 6, 7, and 8, but referring to $H \downarrow_i \Phi$ rather than $H \downarrow_e \Phi$.

Beyond Partial Meet

As in the case for e-contraction we argue that while partial meet Horn i-contractions are all appropriate forms of i-contraction, they do not represent all rational forms of i-contraction. The argument against maxichoice Horn i-contraction is essentially the same one put forward against maxichoice Horn e-contraction. That is, the result $H -_i \Phi$ of maxichoice Horn i-contraction frequently results in a belief set which differs semantically from H by adding a single valuation to the models of H in order to avoid inconsistency. We can, in fact, use a variant of the same example used against maxichoice Horn e-contraction. Let $H = Cn(\{p, q\})$ and $\Phi = \{p \rightarrow \bot\}$. Then $[H] = \{11\}$, the i-remainder sets of Φ w.r.t. H are $H' = Cn(\{p \rightarrow q, q \rightarrow p\})$ and $H'' = Cn(\{q\})$, and $[H'] = \{11, 00\}$ and $[H''] = \{11, 01\}$: i.e., the models of H' and H'' are obtained by adding to the models of H a single valuation in order to avoid inconsistency. The case against partial meet Horn i-contraction is again based on the fact that it does not always include all belief sets between some maxichoice Horn i-contraction and full meet Horn i-contraction, leading us to infra i-remainder sets.

Definition 12 (Infra i-Remainder Sets) *For belief sets H and X, $X \in H \Downarrow_i \Phi$ iff there is some $X' \in H \downarrow_i \Phi$ s.t. $(\bigcap H \downarrow_i \Phi) \subseteq X \subseteq X'$. We refer to the elements of $H \Downarrow_i \Phi$ as the* infra i-remainder sets *of H w.r.t. Φ.*

Horn i-contraction is defined i.t.o. infra i-remainder sets:

Definition 13 (Horn i-Contraction) *An infra i-selection function τ is a function from $\mathscr{P}(\mathscr{P}(\mathcal{L}_\mathsf{H}))$ to $\mathscr{P}(\mathcal{L}_\mathsf{H})$ s.t. $\tau(H \Downarrow_i \Phi) = H$ whenever $\Phi \models \bot$, and $\tau(H \Downarrow_i \Phi) \in H \Downarrow_i \Phi$ otherwise. A contraction function $-_\tau$ is a Horn i-contraction iff $H -_\tau \Phi = \tau(H \Downarrow_i \Phi)$.*

A Representation Result

Our representation result for i-contraction is very similar to that for e-contraction and Postulates $(H{-_i}1)$–$(H{-_i}7)$ below are clearly close analogues of $(H -_e 1)$–$(H -_e 7)$.

$(H -_i 1)$ $H -_i \Phi = \mathit{Cn}(H -_i \Phi)$

$(H -_i 2)$ $H -_i \Phi \subseteq H$

$(H -_i 3)$ If $H + \Phi \not\models \bot$ then $H -_i \Phi = H$

$(H -_i 4)$ If $\Phi \not\models \bot$ then $(H -_i \Phi) + \Phi \not\models \bot$

$(H -_i 5)$ If $\mathit{Cn}(\Phi) = \mathit{Cn}(\Psi)$ then $H -_i \Phi = H -_i \Psi$

$(H -_i 6)$ If $\varphi \in H \setminus (H -_i \Phi)$, there is a H' s.t. $\bigcap(H \downarrow_i \Phi) \subseteq H' \subseteq H$, $H' + \Phi \not\models \bot$, and $H' + (\Phi \cup \{\varphi\}) \models \bot$

$(H -_i 7)$ If $\models \Phi$ then $H -_i \Phi = H$

Analogously with e-contraction, rule $(H{-_i}3)$ can be shown to follow mainly from $(H -_i 6)$. We show that Horn i-contraction is characterised precisely by these postulates.

Theorem 3 *Every Horn i-contraction satisfies $(H -_i 1)$–$(H -_i 7)$. Conversely, every contraction function satisfying $(H -_i 1)$–$(H -_i 7)$ is a Horn i-contraction.*

Package Horn Contraction

The third and last type of contraction we consider is referred to as *package contraction*, a type of contraction studied by Fuhrmann and Hansson (1994) for the classical case (i.e., for logics containing full propositional logic). The goal is to remove *all* sentences of a set Φ from a belief set H. For full propositional logic this is similar to contracting with the disjunction of the sentences in Φ. For Horn logic, which does not have full disjunction, package contraction is more interesting. Our primary interest in package contraction relates to an important version of contraction occurring in ontological reasoning, as we shall see below.

Our starting point is again in terms of remainder sets.

Definition 14 (Horn p-Remainder Sets) *For a belief set H, $X \in H \downarrow_p \Phi$ iff (i) $X \subseteq H$, (ii) $\mathit{Cn}(X) \cap \Phi = \emptyset$, and (iii) for every X' s.t. $X \subset X' \subseteq H$, $\mathit{Cn}(X') \cap \Phi \neq \emptyset$. We refer to the elements of $H \downarrow_p \Phi$ as the* Horn p-remainder sets *of H w.r.t. Φ.*

It is easily verified that Horn p-remainder sets are belief sets. In addition, the following definition will be useful.

Definition 15 *A set Φ is* tautologous *iff for every valuation v, there is a $\varphi \in \Phi$ such that $v \Vdash \varphi$.*

It can be verified that $H \downarrow_p \Phi = \emptyset$ iff Φ is tautologous. (Note that tautologous is not the same as tautological.)

The definition of p-remainder sets is similar enough to that of e-remainder sets (Definition 5) that we can define partial meet Horn p-selection functions, partial meet Horn p-contraction, maxichoice Horn p-contraction, and full meet Horn p-contraction by repeating Definitions 6, 7, and 8, but referring to $H \downarrow_p \Phi$ rather than $H \downarrow_e \Phi$.

Since e- and p-contraction coincide for contraction by singleton sets, our argument also holds for p-contraction. Also, Example 2 is also applicable to p-contraction, from which it follows that partial meet p-contraction is not sufficient either. Consequently, as we did for e-contraction and i-contraction, we move to infra p-remainder sets.

Definition 16 (Infra p-Remainder Sets) *For belief sets H and X, $X \in H \Downarrow_p \Phi$ iff there is some $X' \in H \downarrow_p \Phi$ s.t. $(\bigcap H \downarrow_p \Phi) \subseteq X \subseteq X'$. We refer to the elements of $H \Downarrow_p \Phi$ as the* infra p-remainder sets *of H w.r.t. Φ.*

Horn p-contraction is then defined in terms of infra p-remainder sets in the obvious way.

Definition 17 (Horn p-contraction) *An infra p-selection function τ is a function from $\mathscr{P}(\mathscr{P}(\mathcal{L}_{\mathsf{H}}))$ to $\mathscr{P}(\mathcal{L}_{\mathsf{H}})$ s.t. $\tau(H \Downarrow_p \Phi) = H$ whenever Φ is tautologous, and $\tau(H \Downarrow_p \Phi) \in H \Downarrow_p \Phi$ otherwise. A contraction function $-_\tau$ is a Horn p-contraction iff $H -_\tau \Phi = \tau(H \Downarrow_p \Phi)$.*

A Representation Result

The representation result for p-contraction is very similar to that for e-contraction, with Postulates $(H -_p 1)$–$(H -_p 7)$ being close analogues of $(H -_e 1)$–$(H -_e 7)$.

Observe that the following definition is used in $(H -_p 5)$.

Definition 18 *For sets of sentences Φ and Ψ, $\Phi \hat{=} \Psi$ iff either both are tautologous, or $\forall v \in \mathcal{V}$, $\exists \varphi \in \Phi$ s.t. $v \Vdash \varphi$ iff $\exists \psi \in \Psi$ s.t. $v \Vdash \psi$.*

This definition describes a notion of set equivalence which is appropriate to ensure syntax independence.

$(H -_p 1)$ $H -_p \Phi = \mathit{Cn}(H -_p \Phi)$

$(H -_p 2)$ $H -_p \Phi \subseteq H$

$(H -_p 3)$ If $H \cap \Phi = \emptyset$ then $H -_p \Phi = H$

$(H -_p 4)$ If Φ is not tautologous then $(H -_p \Phi) \cap \Phi = \emptyset$

$(H -_p 5)$ If $\Phi \hat{=} \Psi$ then $H -_p \Phi = H -_p \Psi$

$(H -_p 6)$ If $\varphi \in H \setminus (H -_p \Phi)$, there is a H' s.t. $\bigcap(H \downarrow_p \Phi) \subseteq H' \subseteq H$, $\mathit{Cn}(H') \cap \Phi = \emptyset$, and $(H' + \varphi) \cap \Phi \neq \emptyset$

$(H -_p 7)$ If Φ is tautologous then $H -_p \Phi = H$

Once more $(H{-_p}3)$ is actually redundant here. We show that these postulates characterise Horn p-contraction exactly.

Theorem 4 *Every Horn p-contraction satisfies $(H -_p 1)$–$(H -_p 7)$. Conversely, every contraction function satisfying $(H -_p 1)$–$(H -_p 7)$ is a Horn p-contraction.*

p-Contraction as i-Contraction

In addition to p-contraction being of interest in its own right, we have a specific interest in the case where Φ contains only *basic* Horn clauses: those with exactly one atom in the head and in the body. Our interest in it is because of its relation to an important version of contraction for ontological reasoning in the $\mathcal{EL}$ family of DLs. Briefly, basic Horn clauses correspond closely to *basic* subsumption statements in the $\mathcal{EL}$ family: statements of the form $A \sqsubseteq B$ where A and B are *concept names*. Its importance stems from the fact that basic subsumption statements are used to *repair the subsumption hierarchy*. A detailed investigation of this form of contraction for the $\mathcal{EL}$ family is beyond the scope of this paper. Here we just show that Horn p-contraction with basic Horn clauses can be represented as a special case of Horn i-contraction. Define i as a function from sets of basic Horn clauses to sets of Horn clauses, such that for any set $\Phi = \{p_1 \rightarrow q_1, \ldots, p_n \rightarrow q_n\}$ of basic Horn clauses, we have $i(\Phi) = \{p_1, \ldots, p_n, q_1 \rightarrow \bot, \ldots, q_n \rightarrow \bot\}$.

Proposition 2 *Let H be a Horn belief set and let Φ be a set of basic Horn clauses. Then $K -_p \Phi = K -_i i(\Phi)$.*

It is worth noting that this result does not hold for the case where Φ includes general Horn clauses.

Related Work

In recent years there has been considerable interest in dealing with inconsistent ontologies represented in description logics (Baader et al. 2003) but for the most part, this has not been presented explicitly as a contraction problem. While there has been some work on *revision* for Horn logics (Eiter and Gottlob 1992; Liberatore 2000; Langlois et al. 2008), the only work of importance on Horn contraction, to our knowledge, is that of Delgrande (2008), and this section is mainly devoted to a discussion of his work.

Delgrande defines and characterises a version of e-contraction which introduces additional structure in the choice of e-remainder sets by placing a linear order on *all* e-remainder sets involving a belief set H (i.e., for all possible Φs). When performing contraction by a set Φ, one is obliged to choose the remainder set of H w.r.t. Φ that is *minimal* w.r.t. the linear order. The additional structure imposed by the use of these linear orders ensures that this kind of e-contraction is actually more restrictive than maxichoice e-contraction, although Delgrande refers to it as maxichoice e-contraction. We shall refer to it as *orderly* maxichoice e-contraction. Delgrande conjectures that orderly maxichoice e-contraction is the appropriate form of e-contraction for Horn logic. Our work is not directly comparable to that of Delgrande since he works on the level of *full* AGM contraction, obtained by also considering the *extended* postulates, whereas we are concerned only with *basic* contraction, and leave the extension to full contraction for future work. Nevertheless, it is clear that an extension to full contraction will involve more than just orderly maxichoice e-contraction.

Delgrande also defines a version of orderly maxichoice i-contraction, but his representation result is in terms of maxichoice i-contraction: he refers to it as singleton i-contraction. He takes a fairly dim view of i-contraction, primarily because of the following result: If $p \rightarrow \bot \in H$ then either $q \in H -_i \{p\}$ or $q \rightarrow \bot \in H -_i \{p\}$ for every atom q. His main concern with this is related to the fact that revision defined in terms of the Levi Identity (i-contraction followed by expansion) will yield a result in which all structure of H, given in terms of Horn clauses, is lost. This means that for him a move to partial meet i-contraction is not the solution either, since any prior structure contained in H will still be lost. We view this objection as somewhat surprising in this context, since Horn contraction is intended to operate on the *knowledge level* in which the structure of the theory from which the belief set is generated is irrelevant. So, while we agree that the formal result on which he bases his objections is a good argument against maxichoice i-contraction (it is closely related to our argument against maxichoice e-contraction above, it does not provide a persuasive argument against partial meet i-contraction. It is worth noting that he does not consider i-contraction in terms of infra i-remainder sets at all. Finally, Delgrande expresses doubts about i-contraction, but our result for p-contraction shows that this is too pessimistic.

Conclusion and Future Work

In this paper we have laid the groundwork for contraction in Horn logic by providing formal accounts of *basic* versions of three types of contraction: e-contraction, i-contraction, and p-contraction. Both e-contraction and i-contraction have previously been studied by Delgrande (2008). We have shown that Delgrande's conjectures about orderly maxichoice contraction being *the* appropriate version for these two forms of contraction were perhaps a bit premature.

Here we focus only on *basic* Horn contraction. For future work we plan to investigate Horn contraction for *full* AGM contraction, obtained by adding the *extended* postulates.

And finally, arguably the most interesting of the three versions of contraction we considered is p-contraction, because of its close links with contraction in DLs, specifically the $\mathcal{EL}$ family of DLs. Consequently, for future work we plan to extend our results for Horn contraction to DLs.

References

Alchourrón, C.; Gärdenfors, P.; and Makinson, D. 1985. On the logic of theory change: Partial meet contraction and revision functions. *J. of Symbolic Logic* 50:510–530.

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P., eds. 2003. *Description Logic Handbook*. Cambridge University Press.

Baader, F. 2003. Terminological cycles in a description logic with existential restrictions. In *Proc. IJCAI*, 325–330.

Delgrande, J. 2008. Horn clause belief change: Contraction functions. In *Proc. KR*, 156–165.

Eiter, T., and Gottlob, G. 1992. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *Artificial Intelligence* 57(2–3):227–270.

Fuhrmann, A., and Hansson, S. 1994. A survey of multiple contractions. *Journal of Logic, Language and Information* 3:39–76.

Gärdenfors, P. 1988. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. MIT Press.

Hansson, S. 1999. *A Textbook of Belief Dynamics*. Kluwer.

Langlois, M.; Sloan, R.; Szörényi, B.; and Turán. 2008. Horn complements: Towards Horn-to-Horn belief revision. In *Proc. AAAI*.

Levi, I. 1977. Subjunctives, dispositions and chances. *Synthese* 34:423–455.

Liberatore, P. 2000. Compilability and compact representations of revision of Horn clauses. *ACM Transactions on Computational Logic* 1(1):131—161.

Schlobach, S., and Cornet, R. 2003. Non-standard reasoning services for the debugging of DL terminologies. In *Proc. IJCAI*, 355–360.

Spackman, K.; Campbell, K.; and Cote, R. 1997. SNOMED RT: A reference terminology for health care. *Journal of the American Medical Informatics Association* 640–644.

Combining Motion Planning with an Action Formalism

Jaesik Choi and **Eyal Amir**
Computer Science Department
University of Illinois at Urbana-Champaign
Urbana, IL 61801
{jaesik,eyal}@cs.uiuc.edu

Abstract

Robotic manipulation is important for real, physical world applications. General Purpose manipulation with a robot (eg. delivering dishes, opening doors with a key, etc.) is demanding. It is hard because (1) objects are constrained in position and orientation, (2) many non-spatial constraints interact (or interfere) with each other, and (3) robots may have multi-degree of freedoms (DOF). In this paper we solve the problem of general purpose robotic manipulation using a novel combination of motion planning and an action formalism (Situation Calculus). Our approach integrates motions of a robot with other actions (non-physical or external-to-robot) to achieve a goal while manipulating objects. It differs from previous, hierarchical approaches in that (a) it considers kinematic constraints in configuration space (CSpace) together with constraints over object manipulations; (b) it automatically generates high-level (logical) actions from a CSpace based motion planning algorithm; and (c) it decomposes a planning problem into small segments, thus reducing the complexity of planning.

Introduction

Algorithms for general purpose manipulations of daily-life objects are still demanding (e.g. keys of doors, dishes in a dish washer and buttons in elevators). It was shown that planning with movable objects is P-SPACE hard (Chen and Hwang 1991; Dacre-Wright, Laumond, and Alami 1992; Stilman and Kuffner 2005). Nonetheless, previous works examined such planning in depth (Likhachev, Gordon, and Thrun 2003; Kuffner and LaValle 2000; Kavraki et al. 1996; Brock and Khatib 2000; Alami et al. 1998; Stilman and Kuffner 2005) because of the importance of manipulating objects. The theoretical analysis gave rise to some practical applications (Alami et al. 1998; Cortés 2003; Stilman and Kuffner 2005; Conner et al. 2007), but general purpose manipulation remains out of reach for real-world-scale applications.

Motion planning algorithms have difficulty to represent non-kinematic constraints despite of its strength in planning with kinematic constraints. Suppose that we want to let a robot push a button to turn a light on. CSpace[1] can represent such constraints. However, the CSpace representation could be (1) redundant and (2) computationally inefficient because CSpace is not appropriate for compact representations. It could be redundant, because it always considers the configurations of all objects beside our interests (i.e. a button and a light). Moreover, mapping such constraints into CSpace would be computationally inefficient, because mapping a constraint among n objects could take $O(2^n)$ evaluations in worst case. Thus, most of motion planning algorithms assume that such mappings in CSpace are encoded.

AI planning algorithms and description languages (e.g. PDDL (McDermott 1998)) have difficulty to execute real-world robots despite of its strength in planing with logical constraints. Suppose that we have a PDDL action for 'push the button' which makes a button pushed and a light turned on. However, the PDDL description could be (1) ambiguous and (2) incomplete (require details). Given a robot with m joints, it is ambiguous how to execute the robot to push the button, because such execution is not given in the description. Instead, it assumes that there is a predefined action which makes some conditions (e.g. a button pushed) satisfied whenever precondition is hold and the action is done.

Both methods solve this problem in different ways. Motion planning algorithms use abstractions to solve this problem. AI plannings use manual encodings. Although abstraction provides solutions in a reasonable amount of time in many applications, abstraction lose completeness. Thus, it has no computational benefit in worst cases. Although AI plannings have no need to search the huge CSpace, it requires manual encodings which are not only error-prone but also computationally inefficient.

We solve this problem with combining a motion planning and an AI planning in a model. We extend our previous framework in PDDL (Choi and Amir 2009) into Situation Calculus which provides logic formalisms. That is, Situation Calculus reflecting kinematic constraints are extracted from a graph constructed by a resolution-complete motion planning algorithm. In detail, our algorithm is composed of three subroutines: (1) extracting a graph from a motion planner, (2) building new actions from abstract actions using the built graph, (3) finding a solution in the built action theory, and (4) decoding it into CSpace.

In detail, our algorithm unifies a general purpose (logical) planner and a motion planner in one algorithm. Our algorithm is composed of three subroutines: (1) extracting

[1] CSpace is the set of all possible configurations

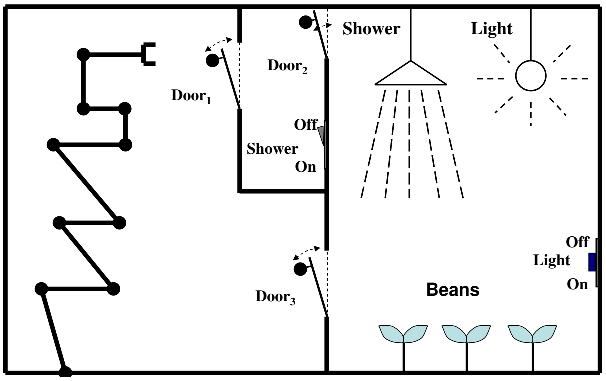

Figure 1: This figure shows an example of manipulating objects with a robotic arm. The goal is to take care of beans in a glasshouse. Beans require water and light everyday. The robot will provide water and light for beans. To accomplish this goal, the arm needs to manipulate objects such as doors and switches.

logical actions from a motion planner, (2) finding an abstract plan from the logical domain, and (3) decoding it into CSpace. It extracts Situation Calculus actions (McCarthy and Hayes 1987; Reiter 2001) from a tree constructed by a motion planner in CSpace. Then, it combines extracted actions with a given BAT (Basic Action Theory explained in Section) that has propositions, axioms (propositional formulae) and abstract Situation Calculus actions. To find an abstract plan efficiently, we automatically partitioned the domain by a graph decomposition algorithm before planning. In the planning step, an abstract plan is found by a factored planning algorithms (Amir and Engelhardt 2003; Brafman and Domshlak 2006) which are designed for the decomposed domain. In decoding, a motion plan is found from the abstract plan.

Section provides a motivational example. Section explains our encoding to build a theory in Situation Calculus. Sections and show our algorithm. Section presents related works. Finally, section provides experimental results followed by the conclusion in section .

A Motivating Example

Figure 1 shows a planning problem. The goal is to provide water and light to beans. The robotic arm should be able to manipulate buttons in the spatial space to provide water and light. There are also non-spatial constraints. At any time either the $shower$ is off or $door_3$ is closed or both.

The planner requires both a general purpose (logical) planner and a motion planner. It requires general purpose planner because the arm needs to revisit some points of CSpace several times in a possible solution. The way points may include '$Open\_door_1$', '$Close\_door_1$', and '$Turn\_light\_on$'. Note that the internal state (values of propositions) can be different, whenever the robot revisits the same point in the CSpace. It is certainly motion planning problem because the kinematic constraints of the arm should be considered. For example, the arm should not collide with obstacles, although the hand of the arm may contact objects.

Hierarchical planners have been classical solutions for these problems. A hierarchical planner takes in charge of high level planning. A motion planner takes in charge of low level planning. However, researchers (or engineers) need to define actions of the robot in addition to axioms among propositions for objects. Without the manual encodings, the hierarchical planner may need to play with the large number of propositions $(O(exp(DOF_{robot}))=|discretized\ CSpace|)$, when DOF_{robot} is the DOF of the robot. With such naive encoding, computational complexity of planning become $(O(exp(exp(DOFs))))$.

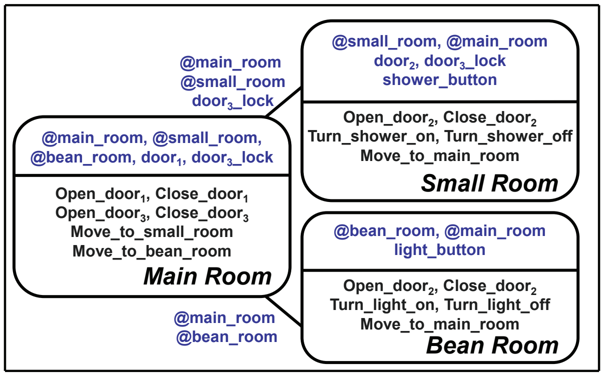

Figure 2: This is a possible tree decomposition for the toy problem of figure 1. The shared propositions appear on edges between subgroups. For example, a proposition ('$@door_3\_lock$') is shared by two subgroups ('Main Room' and 'Small Room') because the proposition is used by actions of two subgroups (respectively '$Open(Close)\_door_3$' and '$Turn\_shower\_on(off)$'). The theory is decomposed into small groups based on the geometric information (eg. the configurations of the room).

Moreover, naive hierarchical planners often have difficulty to find solutions for the following reason. Firstly, it requires interactions between subgoals. For example, the arm must go into the "Bean room" and turns the "light" on (subgoal) before it goes into the "small room" and turns the "shower" on (subgoal). This is essentially the *'Susman anomaly'* which means that the planner dose one thing (being in the Bean room) and then it has to retract it in order to achieve other goal (turning the shower on). Thus, it may require several backtrackings in planning. Secondly, there are two ways of (in principle) achieving "on(light)": (1) going through the small room; and (2) opening door to the Bean room from the Arm-base room. Unless manual encoding is given by an engineer, The latter way (going through the small room) is fine from the perspective of hierarchical planning. However, it will not work in practice because the arm is not long enough (kinematics). Formally, there is no *downward solution*.

Thus, this toy problem shows that (1) hierarchical planning does not work with a naive (simple) encoding, and (2) a complete encoding is too complex to encode manually. We are interested in general principles that underlie a solution to this problem.

In motion planning literature, hybrid planners are used to address these issues (Alami, Siméon, and Laumond 1989; Alami, Laumond, and Siméon 1997; Alami et al. 1998;

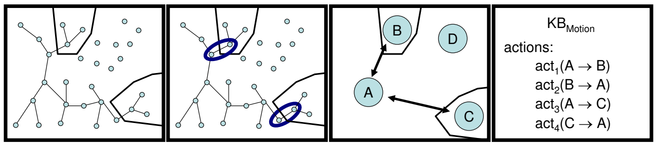
B
D
A
C
$\mathrm{KB}_{\mathrm{Motion}}$
actions:
$\mathrm{act}_1(A \rightarrow B)$
$\mathrm{act}_2(B \rightarrow A)$
$\mathrm{act}_3(A \rightarrow C)$
$\mathrm{act}_4(C \rightarrow A)$

Figure 3: This figure illustrates a process to encode a motion plan into ATM (Action Theory with Motion). The process is follows: (1) a motion plan (a tree) is built by a motion planning algorithm; (2) actions which changes the states of objects are found; (3) propositions are generated (and grouped) based on the found actions; and (4) a ATM is created. Here, we assume that we have a function which provides discrete states of objects given the configuration of an object in finding actions (2). In this figure, the $door_1$ in figure 1 and 2 is closed in a set of states (A). The $door_1$ is moved little in B. However, the $door_1$ is not fully opened. Thus, configurations in the area D is not connected. The area C corresponds to the pushed light button on figure 1 and 2.

Conner et al. 2007; Plaku, Kavraki, and Vardi 2008). However, these are either hard to build due to manual encodings, or infeasible to conduct complex tasks due to the curse of dimensionality of expanded CSpace. The size of CSpace of a hybrid planner exponentially increases with additional movable objects and given propositions. Thus, solving a complex problem may require extensive searches.

Here, we seamlessly combine the general purpose planning and the motion planning. Our planner finds all researchable locations and possible actions that change states of object, states of propositions, or the reachable set of objects.[2] Thus, high-level planner can start to plan based on actions extracted by a motion planner.[3]

However, the number of actions and states can be still intractable. To solve this problem, we partition the domain into the smaller groups of actions and states. For example, the domain can be partitioned as shown in figure 2. It is composed of three parts: (1) operating the shower switch; (2) operating the light switch; and (3) operating in between. The partition can be automatically done with approximate tight bound (Becker and Geiger 1996; Amir 2001).

A factored planner (Amir and Engelhardt 2003) efficiently finds a plan with the partitioned domain. The partitioned groups are connected as a tree shape. In each partitioned domain, our factored planner finds all the possible effects of the set of actions in each factored domain. Then, the planner passes the planned results into the parent of the partition in the tree. In the root node, all the valid actions and effects are gathered. The planner finds a plan for the task, if it exists.

Then, we use a local planner to find a concrete path in CSpace at the final step. However, there is no manual (explicit) encoding (eg. 'turning the *switch A*') between two layers, except logical constraints and mapping functions provided as input.

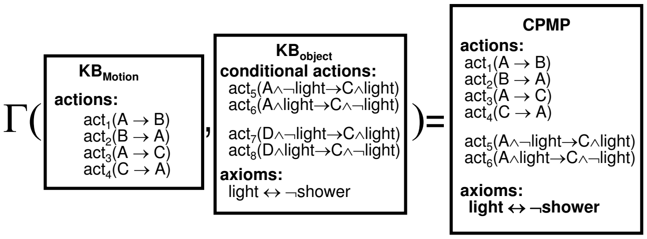
Γ(
$\mathrm{KB}_{\mathrm{Motion}}$
actions:
$\mathrm{act}_1(A \rightarrow B)$
$\mathrm{act}_2(B \rightarrow A)$
$\mathrm{act}_3(A \rightarrow C)$
$\mathrm{act}_4(C \rightarrow A)$
,
$\mathrm{KB}_{\mathrm{object}}$
conditional actions:
$\mathrm{act}_5(A \wedge \neg light \rightarrow C \wedge light)$
$\mathrm{act}_6(A \wedge light \rightarrow C \wedge \neg light)$
$\mathrm{act}_7(D \wedge \neg light \rightarrow C \wedge light)$
$\mathrm{act}_8(D \wedge light \rightarrow C \wedge \neg light)$
axioms:
$light \leftrightarrow \neg shower$
)=
CPMP
actions:
$\mathrm{act}_1(A \rightarrow B)$
$\mathrm{act}_2(B \rightarrow A)$
$\mathrm{act}_3(A \rightarrow C)$
$\mathrm{act}_4(C \rightarrow A)$
$\mathrm{act}_5(A \wedge \neg light \rightarrow C \wedge light)$
$\mathrm{act}_6(A \wedge light \rightarrow C \wedge \neg light)$
axioms:
$light \leftrightarrow \neg shower$

Figure 4: This shows an operation (or algorithm) to combine the extracted ATM with pre-existing BAT. BAT is independently given in a general form to a robot. Thus, ATM can be reusable for robots with different configurations space. Meanwhile, BAT is specific to a robot. Thus, some actions (e.g. act_7 and act_8) in BAT are invalidated, thus excluded in BAT.

Problem Formulation

Preliminaries

Situation Calculus The Situation Calculus is an action formalism which describes the precondition and the effect of each action with First-Order predicate logic formulae. We describe desirable constraints, if it is represented by First-Order formulae.

The Situation Calculus (McCarthy and Hayes 1987; Reiter 2001) is a sorted first order language for representing domains by means of actions, situations, and fluents. Actions and situations are first order terms, and situation-terms stand for history of actions, compound with a function symbol *do*: *do(a, s)* return the situation obtained by executing the action *a* in a situation *s*, which is a sequence of actions.

The dynamic domain is described by a Basic Action Theory BAT = (Σ, D_{S0} , D_{ss}, D_{una}, D_{ap}). Σ includes a set of foundational axioms for situations. D_{S0} is a set of first-order sentences that are valid in S_0. D_{ssa} is a set of successor state axioms for functional and relational fluents. D_{una} is the set of unique names axioms for actions. D_{ap} is a set of action precondition axioms. Please, refer the (Reiter 2001) for detail.

Configuration Space Given a robot and objects, a configuration describes the pose of the robot (r) and objects (O). Configuration space, $CSpace$ is the set of all possible configurations.

The set of configurations that has no collision with obstacles is called the free space C_{free}. The complement of C_{free} in C is called the obstacle region.

Here, we use a sampling-based motion planner (e.g. PRM (Kavraki et al. 1996) or RRT (Kuffner and LaValle 2000)) which extracts a connectivity graph among sampled configurations.

Combining Planning and Motion Planning

Inputs of problem is described as follows.

- $CSpace$: The configuration space of the robot and objects.
- BAT: The Basic Action Theory regarding to actions of the robot over objects.

[2] Here, we assume that we know states of objects without uncertainty as in (Conner et al. 2007).

[3] Our planner may have more actions and states than the hand-encoded case.

- C_{init}: The initial configuration of the robot and objects.
- *Goal*: A first-order formula describing the goal condition.
- *Shared Fluents*: Predicates and functions shared by the Situation Calculus and the CSpace.

Here, if CSpace has n-dimensions, CSpace is $C_1 \times C_2 \times \ldots \times C_n$. We can represent $CSpace$ as $C_{Shared} \times C_{Motion}$. C_{Shared} is the cross product of dimensions which become inputs to BAT of Situation Calculus. C_{Motion} is the cross product of dimensions which are used only in motion planning. Thus, some dimensions become an input to BAT of the Situation Calculus. For theses dimensions, we write Fluents as follows.

$$P(c) \equiv P'(s)(c \in C_{Shared})$$
$$f(c) = f'(s)(c \in C_{Shared})$$

P and P' are predicates. f and f' are functions. P and f include dimensions of CSpace, although P' and f' replace the dimensions with a situation s. c is a configuration. Thus, such predicates and functions relate configurations in CSpace with situations in Situation Calculus.

Definition D_{map}: is defined with following axioms. With a configuration c and a situation s, we build a predicate as follows

$$P_{consistent}(c, s) \equiv \left(\bigwedge_i P_i(c) = P'_i(s) \right)$$

When i is the index for each predicate. In addition, we call the set of axioms with two Fluents as D_{map}.

Thus, our goal is to find a path which achieves the goal conditions while satisfying the action theory and avoiding collision in CSpace.

Combining Planning and Motion Planning (CPMP)

Action Theory with Motion (ATM)

Definition D_{motion}: We build a new theory based on the graph extracted from a motion planning algorithm. A resolution-complete motion planner builds a connectivity graph in CSpace. Based on the graph (V, E), we build two Predicates as follows.

$$\begin{aligned} P_{free}(c) &\equiv \top \textit{ if } c \in V \\ P_{free}(c) &\equiv \bot \textit{ otherwise} \end{aligned}$$

$$\begin{aligned} P_{move}(c, c') &\equiv \top \textit{ if } (c, c') \in E \\ P_{move}(c, c') &\equiv \bot \textit{ otherwise} \end{aligned}$$

$$\begin{aligned} & P_{stable}(c, c') \leftrightarrow \\ & (\forall s (P_{consistent}(c, s) \leftrightarrow P_{consistent}(c', s)) \\ & \wedge (P_{move}(c, c') \vee \exists c'' (P_{stable}(c, c'') \wedge P_{stable}(c'', c')))) \end{aligned}$$

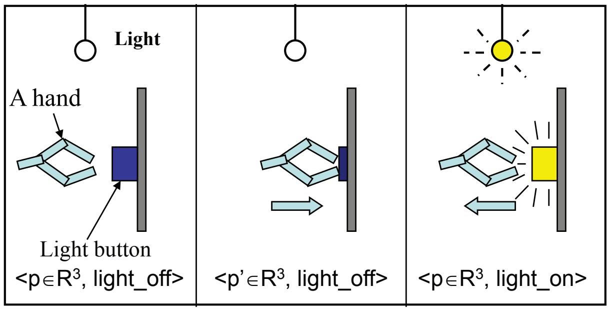

Figure 5: This example shows a situation in which one position in the workspace can correspond to two different states in the combined space (CPMP). Although the physical locations of the arm and button are same in the workspace, an internal state (eg. light is on) is different. The situation can be represented when CSpace and state space in KB are combined (CPMP), even though it is not possible to represent in the classical CSpace alone.

$$\forall c (P_{stable}(C_{init}, c) \rightarrow P_{realize}(c, S_0))$$

C_{init} is the initial configuration. S_0 is the initial situation in the BAT.

We call these sets of predicates as D_{motion}.

Definition D^*_{ap}: We change the set of precondition axioms in D_{ap}. Suppose that we have a following precondition axiom for an action *act* in a situation s.

$$Poss(act, s) \equiv \varphi_{poss}$$

We change it into following

$$\begin{aligned} & Poss_{motion}(act, s) \equiv \varphi_{poss} \\ & \wedge \exists c, c' (P_{realize}(c, s) \wedge P_{consistent}(c', do(act, s)) \\ & \wedge P_{move}(c, c') \wedge P_{free}(c) \wedge P_{free}(c')) \end{aligned}$$

We call the set of modified precondition axioms as D^*_{ap}.

Definition D^*_{eff}: We add the set of effect axioms in D_{eff}. Suppose that we have a following effect axiom for an action *act* in a situation s.

$$Poss(act, s) \rightarrow \varphi_{eff}$$

We change it into following axiom.

$$\begin{aligned} & Poss_{motion}(act, s) \rightarrow \varphi_{eff} \\ & \wedge (\forall c, c', c'' (P_{realize}(c, s) \wedge P_{consistent}(c', do(act, s))) \\ & \rightarrow (P_{realize}(c', do(act, s)) \\ & \wedge (P_{stable}(c', c'') \rightarrow P_{realize}(c'', do(act, s))))) \end{aligned}$$

We call the set of added effect axioms as D^*_{eff}.

We define the unified actions theory, (Combining Planning and Motion Planning)

$$CPMP = (\Sigma, D_{S0}, D_{ss}, D_{una}, D_{motion}, D_{map}, D^*_{ap}, D^*_{eff})$$

Lemma 1. *The complexity of planning problem in the $CPMP$ is as hard as P-SPACE.*

Proof. Any motion planning problem (P-SPACE hard) with movable objects can be reduced to a planning problem in $CPMP$. Suppose that $CPMP$ includes only external propositions which are extracted from the motion planning algorithm. □

Building Actions

We register an action (an edge between two points extracted from a motion planner) into CPMP in when two points have different states in CPMP with regard to mapping function as shown in figure 3. We validate abstract PDDL actions which are realized by the action. Thus, we build a hypergraph whose nodes are sets of modes (CSpace) which have the same state in terms of mapping functions and reachable objects. Our algorithm extensively searches actions with a *resolution complete* motion planner (i.e. PRM) until no new action is found in the hypergraph given a specified resolution.

Lemma 2. *The size of the discretized CSpace for a robot manipulating n objects with given propositions in CPMP is bounded by $O(exp(|objects| + n + p))$, when $|objects|$ is the number of objects, n is the DOF (Degree of Freedom) of the robot, and p is the number of propositions.*

Lemma 3. *The number of possible actions (edges) in the discretized CSpace for objects is only bounded by $O((|objects|) \cdot exp(|objects|))$, when the robot moves one object with an action.*

Proof. From a point in CSpace of object $O(exp(|objects|))$, we can choose an object $O(|objects|)$ to change states. □

Finding a Solution in $CPMP$

We provide a naive algorithm that solves a task in CPMP. Then, we provide two improvements: (1) that solves the problem in the (smaller) factored KBs; and (2) that reduces the number of propositions in $CPMP$ using workspace.

A Naive Solution

Given a task of CPMP, *NaiveSolution* finds a solution. It may use a general purpose planner ($GeneralPlanner$) to find an abstract solution. Then, ($LocalMotionPlan$) encodes a path in CSpace.

```
Algorithm:NaiveSolution
Input: $r$(a robot), $BAT$(Basic Action Theory), $s_{start}$(initial
       state), and $s_{goal}$(goal condition)
Output: $path_{concrete}$(Solution)
$ATM \leftarrow$ FindActionFromMP($r$)
$CPMP = \Gamma(ATM, BAT)$
$path_{abstract} \leftarrow$ GeneralPlanner( $CPMP$, $s_{start}$, $s_{goal}$)
$path_{concrete} \leftarrow$ LocalMotionPlan( $path_{abstract}$ )
```

Algorithm 1: *NaiveSolution* provides a path for a robot. It uses a general planner ($GeneralPlanner$) to find an abstract solution. Then, it is encoded into the path in the CSpace by a motion plan ($LocalMotionPlan$).

Tree Decomposition of KB with Objects

Given a KB, finding a tree-decomposition of the minimum treewidth is a NP-hard problem. However, the complexity is only bounded by the treewidth of CPMP, if a tree-decomposition is found by an efficient heuristic (Becker and Geiger 1996; Amir 2001).

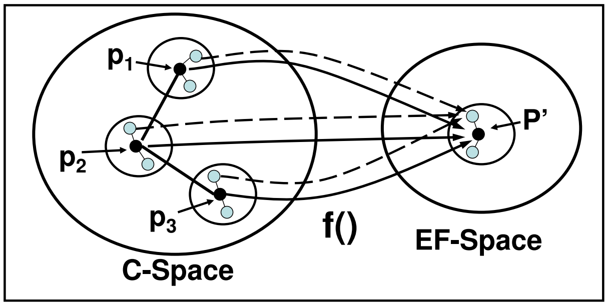

Figure 6: This figure shows a mapping function (f()) from a CSpace to an EF-Space. p_1, p_2, and p_3 in CSpace are mapped into p' in EF-Space. The connected lines ((p_1, p_2) and (p_2, p_3)) represent the first condition of Theorem 3. The circles represent the second condition.

Theorem 4. *The complexity of planning in CPMP is bounded by $O(exp(tw(CPMP)))$ if the tree-decomposition is given.*[4]

Proof. Proofs in (Brafman and Domshlak 2006; Amir 2001) can be easily modified to prove this theorem. □

From Exponential CSpace to Polynomial EF-Space

In this section, we provide a generalized method which project CSpace into much smaller workspace. It is an extension of our previous work (Choi and Amir 2007). it efficiently finds a solution when the projection method is applicable. Here, we want to transform CSpace into a smaller space, EF-Space, using a mapping function $f()$. The function ($f()$) maps each point (p) in CSpace into a point (p') in EF-Space with satisfying following conditions.

1. When P is a set of points whose image are p' in EF-Space ($f(p) = p'$), any pair of two elements (p_1, $p_2 \in P$) is connected each other in CSpace;
2. When two points (p and q) are mapped into two points (p' and q') in EF-Space. p and q are connected neighbor if and only if p' and q' are connected neighbor.

Two points are connected neighbor means when they are directly connected in the space.

Theorem 5. *The complexity of motion planning in EF-Space is bounded by following*

$$O(\textit{EF-Space}) \cdot O(max_{ep \in \textit{EF-Space}}(ball(P_{ep}))).$$

P_{ep} is a set of points whose image is ep. (That is, $P_{ep} = \{p|f(p) = ep\}$) The $ball(P)$ is volume of the ball which includes P.

Proof. Given a motion planning problem (an initial configuration and goal one), a path in EF-Space can be found in O(EF-Space) with a graph search algorithm. Given the path in EF-Space, one needs to search the whole ball in worst case. □

One simple example of EF-Space is the workspace of end-effector. Suppose that the points in CSpace are mapped into

[4] tw(KB) is the treewidth of KB.

the points of end-effector in workspace. One can build an algorithm that finds all the neighboring points from the innermost joint (or wheel) to the outermost joint with a dynamic programming. If points of the previous joint are connected to all neighboring points, the neighboring points of the current joint are found by a movement of current joint (current step) or a movement of any previous joint (previous steps). The found connected points in workspace satisfy the second conditions, if the first condition holds in the workspace.

In worst case, the first condition is hard to satisfy. In the environment, the mapping function (f) should be bijective. Thus, the EF-Space is nothing but the CSpace. However, the first condition holds in many applications where the distance between obstacles (or objects) and the robot is far enough. That is the theoretical reason why the planning problem in the sparse environment is easy even in CSpace.

Moreover, one can find another EF-Space considering topological shape of robot (Choi and Amir 2007). In the space, two points (p_1 and p_2) are mapped into the same point p'_1 if two configurations (p_1 and p_2) are homotopic, and they indicate the same end point. Otherwise, another point p'_2 is generated in the EF-Space. In 2D, an island obstacle divides configurations into two groups for each side (left or right). Thus, the EF-Space is exponentially proportional to the number of island obstacles. However, EF-Space itself is bounded by the workspace whose size is polynomial to the number of joints. Thus, it is much smaller than the CSpace and rather larger than the workspace.

A Unified Motion Plan

We present our algorithms in this section. The main algorithm , *UnifiedMotionPlanner* (Algorithm 2), is composed of three parts: *FindActionFromMP* (Algorithm 3); *FactoredPlan* (Algorithm 4); and *LocalPlanner*. The goal of *UnifiedMotionPlanner* is to find a solution to achieve a goal situation.

```
Algorithm:UnifiedMotionPlanner
Input: $r$(a robot), $BAT$(Basic Action Theory), $s_{start}$(initial
    state), $s_{goal}$(goal condition)
Output: $path_{concrete}$(Solution)
$ATM \leftarrow$ FindActionFromMP($r$)
$CPMP = \Gamma(ATM, BAT)$
$KB_{Tree} \leftarrow$ PartitionKBtoTree($CPMP$)
$path_{abstract} \leftarrow$ FactoredPlan( $KB_{Tree}$, $s_{start}$, $s_{goal}$ )
$path_{concrete} \leftarrow$ LocalPlan( $path_{abstract}$ )
return $path_{concrete}$
```

Algorithm 2: *UnifiedMotionPlaner* finds all the reachable locations and actions in each location with FindActionFromMP. A motion planner is embedded in FindActionFromMP to extract abstracted actions in CSpace. Then, PartitionKBtoTree partitions the $CPMP$ into a tree. FactoredPlan finds a solution given the pair of initial and goal condition in the partitioned tree domain. The LocalPlan finds a concrete path for the robot.

FindActoinFromMP

FindActionFromMP searches all the reachable locations and actions in CSpace or EF-Space. In both cases, it has a dramatically reduced space.

```
Algorithm:FindActionFromMP
Input: $r$(a robot)
Output: $ATM$(extracted actions)
$MP_{Tree} \leftarrow$ a random tree in CSpace built by a motion
planner (e.g. Probabilistic Roadmap, Factored-Guided Motion
Planning)
for each edge ($e_{ij}$) $\in MP_{Tree}$ do
    if $state(p_i) \neq state(p_j)$ then
        $KB_M \leftarrow KB_M \bigcup D^*_{ap}$ (as in section
        $KB_M \leftarrow KB_M \bigcup D^*_{eff}$ (as in section
return $KB_M$
```

Algorithm 3: . *FindActionFromMP* finds all abstract actions for a robot. A motion planner (eg. *FactorGuidedPlan* or *RoadmapMethod*) recursively finds all the reachable locations and actions. Then, the algorithm insert actions of each configuration (c_{ij}) of objects in the workspace. It assume that the object is in the configuration (c_{ij}). Thus, the condition (configuration of objects) is combined into the actions (act_{ij}). The union of all actions becomes the KB_M.

FactoredPlan

FactoredPlan finds a solution after factoring the domain (the space of end-effector in workspace) into small domains. It decomposes the domain into a tree in which each partitioned group becomes nodes, and shared axioms appear on a link between nodes. Then, it finds partial plans for a node and its children nodes with assuming that the parents nodes may change any shared states in between. After all, it finds a global solution in the root node.

```
Algorithm:FactoredPlan
Input: $KB_{Tree}$ (partitioned KB as a tree), $s_{start}$ (initial
    states), $s_{goal}$ (goal condition)
Output: $path_{abstract}$ (An abstract plan)
depth $\leftarrow$ (predefined) number of interaction between domains.
for each node($KB_{part}$) in $KB_{Tree}$ from leaves to a root do
    $Act_{ab} \leftarrow$ PartPlan( $KB_{part}$, depth) .
    SendMessage( $Act_{ab}$, the parent node of $KB_{part}$ )
$path_{ab} \leftarrow$ a solution from $s_{init}$ to $s_{goal}$ in the root node of
$KB_{tree}$
return $path_{ab}$
```

Algorithm 4: *FactoredPlanning* algorithm automatically partitions the domain to solve the planning problem (from s_{init} to s_{goal}). It iterates domains from leaves to the root node without backtracks. In each node, *PartPlan* finds all possible actions that change shared states in the parents node. *PartPlan* assumes that the parent node may change any states in the shared states in between. The planned actions in the subdomain become an abstract action in the parent node. They are sent by *SendMessage*.

Related Works

Here, we review the related works in two aspects: (1) using logical representation in robot planning; and (2) modifying the motion planning algorithm to achieve complex task (eg.

manipulating objects). One may see the former way as top-down and the latter way as bottom-up.

(Alami et al. 1998) presents a well-integrated robot architecture which controls multiple robots. It uses logical representations in higher level planners and CSpace based motion planners in lower-level planning. However, the combination of two planners is rather naive (manual).

Recently, (Conner et al. 2007) provides an improved way to combine the *Linear Temporal Logic (LTL)* to control continuously moving cars in the simulated environment.[5] However, their model is a nondeterministic automata, while our model is deterministic. Due to the intractability of nondeterministic model, their representation is restricted to a subset of LTL to achieve a tractable (polynomial time) algorithm. Experiments are focused on controlling cars instead of manipulating objects.

Motion planning research has a long-term goal of building a motion planning algorithm that finds plans for complex tasks (eg. manipulating objects). (Stilman and Kuffner 2005) suggests such a planning algorithm based on a heuristic planner (Chen and Hwang 1991) which efficiently relocates obstacles to reach a goal location. Recently, it was extended to embed constraints over objects into the *CSpace* (Stilman 2007). In fact, the probabilistic roadmap method (Kavraki et al. 1996) of the algorithm is highly effective in manipulating objects. However, we argue that our algorithm (factored planning) is more appropriate in terms of generality and efficiency than a search-based (with backtracks) heuristic planner.

Other works also make efforts in this direction to build a motion planning algorithm for complex tasks. (Plaku, Kavraki, and Vardi 2008) solves a motion planning problem focused on safety with logical constraints represented with LTL . (M. Pardowitz 2007) focuses on learning actions for manipulating objects based on the explanation based learning (Dejong and Mooney 1986). They use a classical hierarchical planner in planning. (J. Van den Berg 2007) provides an idea that extracts the propositional symbols from a motion planner. The symbols are used to check the satisfiability of the planning problems. (S. Hart 2007) uses a potential field method to achieve complex tasks with two arms. However, the main interests of these works are not planning algorithm, or are limited to the rather simpler tasks.

An Experiment in Simulation

We build our algorithm for a task that pushes buttons to call numbers. There are 8 buttons in total. 4 buttons ($key1(P1)$, $key2(P2)$, $unlock(P3)$, and $lock(P4)$) are used to lock (and unlock) the buttons. Other 4 buttons ($\#A(P5)$, $\#B(P6)$, $\#C(P7)$ and $Call(P8)$) are used to make phone calls. Initially, the button is locked, the robot needs to push unlock buttons after pushing both key buttons ($P1$ and $P2$). Then, the robot can make a phone call with pushing the $Call$ button ($P8$) after selecting an appropriate number among $\#A(P5)$, $\#B(P6)$, and $\#C(P7)$. After a call, the buttons are automatically unlocked. We encode such constraints and action in a PDDL.[6]

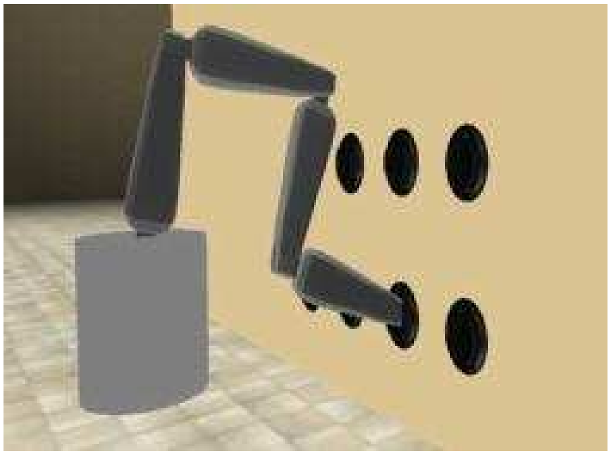

Figure 7: This is a capture of the motion of push button in the wall in experiments. The robot has 5 DOFs (rotational joints on the base and 4 revolute joints on the arm). We do experiment with increasing the number of joints from 2 to 9.

We build a tree from a randomized algorithm with 80000 points in CSpace. With a labeling function that returned the states of buttons, we found 33 edges in the tree[7]. They are encoded into 8 actions for 8 buttons. Then, the combined KB ($CPMP$) is used to find a goal (calling all numbers ($\#A$, $\#B$, and $\#C$). The returned abstract actions are decoded into a path on the tree of motion plan. Figure 7 is a snapshot of the simulation.[8]

In this experiment, we focus on extracting actions from a motion planning algorithm, because the factored planer itself is not a contribution of this paper. Theoretical and experimental benefits of *FactoredPlan* is shown in the previous papers (Amir and Engelhardt 2003; Brafman and Domshlak 2006). We run our simulation on a general purposed planner (Fourman 2007). Thus, the *NaiveSolution* algorithm is used in this simulation.

Conclusions and Future Research

We present an algorithm that combines the general purpose (logical) planner and a motion planner. Our planner is designed to manipulate objects with robot. To solve the problem, previous works used a hierarchical planner (high-level) and a motion planner (low-level). Most of them used manual encodings between two layers. That was one of technical hardness of this problem.

Theoretically, combining such planners is hard for the following reasons: (1) hierarchical planner is hard and not feasible sometime; and (2) direct combination of CSpace and state space gives an doubly exponential search problem.

[5] Any *First Order Logic (FOL)* sentences can be reduced to *Linear Temporal Logic (LTL)*. Thus, LPL is a superset of FOL.

[6] Situation Calculus encoding is not impleted yet

[7] We simplify the manipulations for attaching and detaching buttons

[8] The details of encoded actions and movies are available at http://reason.cs.uiuc.edu/jaesik/cpmp/supplementary/.

Moreover, we can loss the geometric motion planning information, if we translate everything to PDDL (McDermott 1998) without a motion planner.

We combine the CSpace and state space in a KB, CPMP (Combining Planning and Motion Planning). Moreover, we provide the computational complexity of the problem. We also argue that the treewidth of CPMP determines the hardness of a manipulation task.

However, the suggested algorithm still has some limitations that need to be improved in future research. The exploration steps in *FindActionFromMP* may take long time due to the large cardinality of state space $(O(n + |objects| + p)$ as in lemma 2. Assumptions of EF-space would inappropriate for cluttered environments where $O(max_{ep \in \text{EF-Space}}(ball(P_{ep}))$ of theorem 5 are intractable.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. 05-46663. We also thank UIUC/NCSA Adaptive Environmental Sensing and Information Systems (AESIS) initiative for funding part of the work.

References

Alami, R.; Chatila, R.; Fleury, S.; Ghallab, M.; and Ingrand, F. 1998. An architecture for autonomy. *International Journal of Robotics Research* 17(4):315–337.

Alami, R.; Laumond, J.-P.; and Siméon, T. 1997. Two manipulation planning algorithms. In Laumond, J.-P., and Overmars, M., eds., *Algorithms for Robotic Motion and Manipulation*. Wellesley, MA: A.K. Peters.

Alami, R.; Siméon, T.; and Laumond, J.-P. 1989. A geometrical approach to planning manipulation tasks. In *Proceedings International Symposium on Robotics Research*, 113–119.

Amir, E., and Engelhardt, B. 2003. Factored planning. In *IJCAI*, 929–935.

Amir, E. 2001. Efficient approximation for triangulation of minimum treewidth. In *UAI*, 7–15.

Becker, A., and Geiger, D. 1996. A sufficiently fast algorithm for finding close to optimal junction trees. In *UAI*, 81–89.

Brafman, R. I., and Domshlak, C. 2006. Factored planning: How, when, and when not. In *AAAI*.

Brock, O., and Khatib, O. 2000. Real-time replanning in high-dimensional configuration spaces using sets of homotopic paths. In *ICRA'00*, 550–555.

Chen, P., and Hwang, Y. 1991. Motion planning for a robot and a movable object amidst polygonal obstacles. In *ICRA'91*, 444–449.

Choi, J., and Amir, E. 2007. Factor-guided motion planning for a robot arm. In *IROS'07*.

Choi, J., and Amir, E. 2009. Combining planning and motion planning. In *ICRA'09*, 238–244.

Conner, D. C.; Kress-Gazit, H.; Choset, H.; Rizzi, A.; and Pappas, G. J. 2007. Valet parking without a valet. In *IROS'07*.

Cortés, J. 2003. *Motion Planning Algorithms for General Closed-Chain Mechanisms*. Ph.D. Dissertation, Institut National Polytechnique de Toulouse, Toulouse, France.

Dacre-Wright, B.; Laumond, J.-P.; and Alami, R. 1992. Motion planning for a robot and a movable object amidst polygonal obstacles. In *ICRA'92*, volume 3, 2474–2480.

Dejong, G., and Mooney, R. 1986. Explanation-based learning: An alternative view. *Mach. Learn.* 1(2):145–176.

Fourman, M. 2007. Propplan. Software.

J. Van den Berg, M. O. 2007. Kinodynamic motion planning on roadmaps in dynamic environments. In *IROS'07*.

Kavraki, L. E.; Svestka, P.; Latombe, J.-C.; and Overmars, M. 1996. Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Trans. on Rob. and Auto.* 12(4):566–580.

Kuffner, J. J., and LaValle, S. M. 2000. RRT-connect: An efficient approach to single-query path planning. In *ICRA'00*.

Likhachev, M.; Gordon, G.; and Thrun, S. 2003. ARA*: Anytime A* search with provable bounds on sub-optimality. In *NIPS'03*.

M. Pardowitz, R. Zollner, R. D. 2007. Incremental acquisition of task knowledge applying heuristic relevance estimation. In *IROS'07*.

McCarthy, J., and Hayes, P. J. 1987. *Some philosophical problems from the standpoint of artificial intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

McDermott, D. 1998. The planning domain definition language manual.

Plaku, E.; Kavraki, L. E.; and Vardi, M. Y. 2008. Hybrid systems: From verification to falsification by combining motion planning and discrete search. *Formal Methods in System Design*.

Reiter, R. 2001. *Knowledge in action : logical foundations for specifying and implementing dynamical systems*. Cambridge, Mass.: MIT Press. The frame problem and the situation calculus.

S. Hart, R. G. 2007. Natural task decomposition with intrinsic potential fields. In *IROS'07*.

Stilman, M., and Kuffner, J. 2005. Navigation among movable obstacles: Real-time reasoning in complex environments. *International Journal of Humanoid Robotics* 2(4):479–504.

Stilman, M. 2007. Task constrained motion planning in robot joint space. In *IROS'07*.

Tractable First-Order Golog with Disjunctive Knowledge Bases

Jens Claßen and **Gerhard Lakemeyer**
Department of Computer Science
RWTH Aachen University
Germany
⟨classen|gerhard⟩@cs.rwth-aachen.de

Abstract

While based on the Situation Calculus, current implementations of the agent control language Golog typically avoid offering full first-order capabilities, but rather resort to the closed-world assumption for the sake of efficiency. On the other hand, realistic applications need to deal with incomplete world knowledge including disjunctive information. Recently Liu, Lakemeyer and Levesque proposed the logic of limited belief $\mathcal{SL}$, which lends itself to efficient reasoning in incomplete first-order knowledge bases. In particular, $\mathcal{SL}$ defines levels of belief which limit reasoning by cases in a principled way. In this paper, we propose to apply $\mathcal{SL}$-based reasoning in the context of a Golog system. Central to our approach is a new search operator that finds plans only within a fixed belief level k, and an iterative-deepening-style variant where instead of considering plans with increasing length, the belief level k is incremented in each cycle. Thus, not the shortest plans are preferred, but those which are the computationally cheapest to discover.

Introduction

The agent language Golog (Levesque et al. 1997) has already been applied in many application scenarios, including the control of autonomous mobile robots (Ferrein and Lakemeyer 2008). The language is based on the Situation Calculus (McCarthy and Hayes 1969; Reiter 2001), which in the theoretical formalization is a dialect of first-order predicate calculus. However, current implementations of Golog typically avoid to offer full first-order capabilities, but rather resort to the closed-world and/or domain closure assumptions for the sake of efficiency of reasoning. On the other hand, in realistic applications such as mobile robotics, almost inevitably one has to cope with incomplete world knowledge, in particular in the form of disjunctive information. Furthermore, as the task of an autonomous robot is usually open-ended, not all individuals (persons or objects) it has or will have to deal with are known in advance.

$\mathcal{SL}$, the subjective logic of limited belief proposed by Liu, Lakemeyer and Levesque (2004) is a formalism for efficient reasoning with incomplete first-order knowledge bases. They define a family of believe operators $\mathbf{B}_0$, $\mathbf{B}_1$, $\mathbf{B}_2$,... where intuitively $\mathbf{B}_0$ corresponds to the agent's explicit belief and implicit beliefs become only available at higher belief levels, where the greater k, the computationally more expensive, roughly measured in terms of the number of nested case distinctions. When k is fixed, then whether $\mathbf{B}_0 KB$ implies $\mathbf{B}_k\phi$ is decidable, and when the KB is in a certain form, reasoning is also tractable. Furthermore, the inference is classically sound in the sense that when $\mathbf{B}_0 KB$ implies $\mathbf{B}_k\phi$ in $\mathcal{SL}$, then KB entails ϕ in classical predicate logic.

For the above mentioned reasons, we believe that it is beneficial to apply $\mathcal{SL}$-based reasoning in the context of a Golog system. Apart from the fact that reasoning within a fixed level k can be done efficiently, belief levels offer the possibility to define a new planning operator that prefers plans with least computational costs. To illustrate the idea, consider a Golog program of the following form:

$$\psi?; a \mid \varphi?; b; c$$

There is a nondeterministic choice (|) between two branches: if formula ψ holds, action a can be executed, or when formula φ holds, action sequence $b; c$ could be performed. Planning here means to resolve the nondeterminism and thus commit to one or the other branch. A classical Golog system will typically test the branches in the presented order, meaning it first checks whether the action sequence $\langle a\rangle$ constitutes a legal execution, which involves checking if ψ is known to hold according to the system's knowledge base. Formula φ and sequence $\langle b, c\rangle$ will only be tested once Golog found out that it is not possible to execute the left branch successfully. Alternatively, the system might apply some iterative deepening strategy, which always yields the shortest action sequences. Still, when the left branch in the above example is executable, the system would prefer it over the right one. In any case, no attention is paid to the computational effort involved.

Now assume that φ can instantly be inferred from the agent's knowledge base (say if it is a fact that is explicitly known to be true), but ψ is quite complicated and requires extensive reasoning. In particular when the decision has to be made quickly (e.g. think of robot soccer) or when the additional reasoning time outweighs the time saved by performing fewer actions, it may pay off to prefer possibly longer plans that however involve less reasoning.

As our running example, consider a mobile robot working in an office environment. There is one employee, Carol, who wants to have a look at a certain book. The department possesses two copies, where one (*book1*) is usually located in the library (*lib*) and the other one (*book2*) in the lab (*lab*). She gives the robot the following orders: "If *book1* is in the library, bring it to me *or* if *book2* is in the lab, bring it after unlocking the lab door." This might be expressed as follows in a Golog program:

$$\begin{aligned}&\mathit{At}(\mathit{book1}, \mathit{lib})?; \mathit{get}(\mathit{book1}, \mathit{lib}) \mid\\ &\mathit{At}(\mathit{book2}, \mathit{lab})?; \mathit{unlock}(\mathit{lab}); \mathit{get}(\mathit{book2}, \mathit{lab})\end{aligned}$$

Now assume that the robot explicitly knows from a recent observation that *book2* actually is in the lab. On the other hand, it only knows that it saw *book1* yesterday in the office shared by Ann and Bob, meaning one of them borrowed it. The robot also knows that whoever borrows a book will return it to the library in the evening on the same day. As working hours have just begun, all books that were borrowed yesterday will now be in the library. Obviously, this knowledge is sufficient to deduce that *book1* is in the library, but whereas retrieving the explicit fact $\mathit{At}(\mathit{book2}, \mathit{lab})$ from the knowledge base basically requires no reasoning at all, deriving $\mathit{At}(\mathit{book1}, \mathit{lib})$ involves one case distinction: Either Ann or Bob borrowed the book, but in any case, it has been returned. Therefore intuitively, $\mathit{At}(\mathit{book2}, \mathit{lab})$ is already available at belief level zero, but $\mathit{At}(\mathit{book1}, \mathit{lib})$ only at greater levels.

In this paper we propose to apply the idea of iterative deepening on belief levels instead of on action sequence lengths. It will first be tested whether any of the program's possible execution traces can be verified to succeed by reasoning at level zero. Only if this is not the case, the level will be increased to one etc. Thus, the first successful execution trace to be found will also be the one that needs the least computational effort, which allows to obtain solutions much quicker in many cases.

The remainder of the paper is organized as follows. In the next section, we give the formal syntax and semantics of the logic on which our approach is based. Next, we present some results that relate the formalism to existing languages. The following section contains the main contribution of this paper in form of the new planning operator we propose. Finally, we sketch possible directions for future work.

Definitions

In this section, we introduce our new logic $\mathcal{SLA}$ formally. The language is basically an extension of Liu, Lakemeyer and Levesque's (2004) logic of limited belief $\mathcal{SL}$ by aspects of the modal Situation Calculus variant $\mathcal{ES}$ (Lakemeyer and Levesque 2004) for modelling action and change.

Syntax

Terms The *terms* of the language come in two sorts: *object* and *action*. A term of sort object is either an object variable ($x_1, x_2, \ldots$) or an object constant ($d_1, d_2, \ldots$, e.g. lab). An action term is either an action variable ($a_1, a_2, \ldots$) or of the form $g(t_1, \ldots, t_n)$, where g is an action function of arity n (e.g. unlock) and the t_i are object terms.

Formulas The *objective formulas* form the least set where

1. any atom of the form $F(t_1, \ldots, t_n)$ is an objective formula, where F is a fluent predicate symbol of arity n (e.g. *At*) and the t_i are object terms;
2. when t_1 and t_2 are terms of sort object, then $(t_1 = t_2)$ is an objective formula;
3. when t is a non-variable term of sort action, x an object variable, and ϕ, ϕ' are objective formulas, then so are $[t]\phi$, $\mathit{Poss}(t)$, $\exists x\phi$, $\neg\phi$, and $\phi \vee \phi'$.

We read $[t]\phi$ as "ϕ holds after doing action t" and $\mathit{Poss}(t)$ as "action t is possible to execute". We further call an objective formula *static* when it does not contain any action terms. Note that we disallow equalities and quantification over actions. The *subjective formulas* form the least set with

1. if ϕ is an objective formula and $k \geq 0$, then $\mathbf{B}_k\phi$ is a subjective formula and called a *believe atom* at level k;
2. if t_1 and t_2 are terms of sort object, then $(t_1 = t_2)$ is a subjective formula;
3. if φ_1 and φ_2 are subjective formulas and x is a variable of sort object, then $\neg\varphi_1$, $(\varphi_1 \vee \varphi_2)$ and $\exists x\varphi_1$ are also subjective formulas.

The language $\mathcal{SLA}$ is the set of all subjective formulas as defined above. Therefore, very much similar to $\mathcal{SL}$, all (fluent) predicates other than equality must occur within the scope of a $\mathbf{B}_k$ operator, which must not be nested. Here, we further require that also $[t]$ and $\mathit{Poss}(t)$ operators do not appear outside of $\mathbf{B}_k$. The fact that we only study formulas talking about the agent's beliefs about the world state is why the language is called *subjective logic*, or in our case, *subjective logic of actions*. $(\varphi_1 \wedge \varphi_2)$, $\forall x\varphi$, $(\varphi_1 \supset \varphi_2)$ and $(\varphi_1 \equiv \varphi_2)$ are treated as the usual abbreviations.

Programs Programs are composed according to the following grammar:

$$\delta ::= t \mid \phi? \mid (\delta_1; \delta_2) \mid (\delta_1|\delta_2) \mid \pi x.\delta \mid \delta^*$$

Here, t is any (not necessarily ground) term of sort action, ϕ can be any objective formula, and x an object variable. In the presented order, the constructs mean a primitive action, a test, sequence of programs, nondeterministic choice between programs, nondeterministic choice of argument, and nondeterministic iteration.

Regression and Basic Action Theories

Before we define the logic's formal semantics, we introduce basic action theories and regression, following (Lakemeyer and Levesque 2004). The language for basic action theories consists of the objective formulas defined above and extended by another modal operator $\Box$, where $\Box\alpha$ reads "α holds after every sequence of actions.", as well as equality atoms $(t_1 = t_2)$ among action terms, where at most one of the t_i is a variable. A formula without $\mathit{Poss}(t)$ and $[t]$, but possibly containing such action equalities, is called *quasi-static*.

Definition 1 (Basic Action Theory) *Given a set of fluent predicates $\mathcal{F}$, a set of sentences Σ is called a* basic action theory *over $\mathcal{F}$ iff it only mentions the fluents in $\mathcal{F}$ and is of the form $\Sigma = \Sigma_0 \cup \Sigma_d$, where $\Sigma_d = \Sigma_{\mathit{pre}} \cup \Sigma_{\mathit{post}}$ and*

- Σ_0 *is a finite set of static sentences,*
- Σ_{pre} *is a singleton of the form* $\Box(Poss(a) \equiv \pi)$*, where* π *is quasi-static with* a *being the only free variable;*
- Σ_{post} *is a finite set of successor state axioms of the form* $\Box(([a]F(\vec{x})) \equiv \gamma_F)$*, one for each fluent* $F \in \mathcal{F}$*, where* γ_F *is a quasi-static formula whose free variables are among* $\vec{x}$ *and* a*.*

In our example, we might have an initial KB Σ_0 containing

$$\begin{array}{ll} At(book2, lab), & \\ Borrowed(ann, book1) \lor Borrowed(bob, book1), & (1) \\ \forall x \forall y Borrowed(x, y) \supset At(y, lib) & \end{array}$$

where $Borrowed(x, y)$ means that person x borrowed y yesterday. The precondition axiom Σ_{pre} is given by:

$$\begin{array}{l} \Box Poss(a) \equiv \; \exists x(a = unlock(x) \lor a = lock(x)) \; \lor \\ \quad \exists x \exists y((a = get(x, y) \lor a = put(x, y)) \land \neg Locked(y)) \end{array}$$

That is locking or unlocking is always possible, but putting or getting something only when the according location is not locked. The successor state axioms in Σ_{post} are

$$\begin{array}{l} \Box[a]At(x, y) \equiv a = put(x, y) \lor At(x, y) \land a \neq get(x, y) \\ \Box[a]Locked(x) \equiv a = lock(x) \lor Locked(x) \land a \neq unlock(x) \end{array}$$

Whereas Lakemeyer and Levesque (2004) provide a complete model-theoretic semantics for $\Box$ and $[\cdot]$ within their logics $\mathcal{ES}$, we here adapt a view similar to (Liu, Lakemeyer, and Levesque 2004), i.e. we are interested in the implicit conclusions an agent can draw, given certain explicit beliefs, and the computational costs for doing so. In our encoding, the precondition and successor state axioms of basic action theories are part of the agent's explicit belief, and conclusions about future situations are drawn using regression.

Regression is a method for computing projections by syntactically transforming a formula talking about future situations (after performing certain actions) into an equivalent formula that only talks about the current situation. We use an adaptation of Lakemeyer and Levesque's $\mathcal{ES}$ variant of Reiter's (2001) regression operator as follows:

Definition 2 (Regression) *Formally, for any objective formula* α*, let* $\mathcal{R}[\Sigma_d, \alpha]$*, the regression of* α *wrt* Σ_d*, be the formula* $\mathcal{R}[\Sigma_d, \langle\,\rangle, \alpha]$*, where for any sequence of action terms* σ *(not necessarily ground),* $\mathcal{R}[\Sigma_d, \sigma, \alpha]$ *is defined inductively on* α *by:*

1. $\mathcal{R}[\Sigma_d, \sigma, (t_1 = t_2)] = (t_1 = t_2)$,
   where the t_i *are object terms;*
2. $\mathcal{R}[\Sigma_d, \sigma, (g_1(\vec{t_1}) = g_2(\vec{t_2}))] = \bot$,
   where g_1 *and* g_2 *are distinct action symbols;*
3. $\mathcal{R}[\Sigma_d, \sigma, (g(\vec{t_1}) = g(\vec{t_2}))] = (\vec{t_1} = \vec{t_2})$*;*
4. $\mathcal{R}[\Sigma_d, \sigma, \neg\alpha] = \neg\mathcal{R}[\Sigma_d, \sigma, \alpha]$*;*
5. $\mathcal{R}[\Sigma_d, \sigma, (\alpha \lor \beta)] = (\mathcal{R}[\Sigma_d, \sigma, \alpha] \lor \mathcal{R}[\Sigma_d, \sigma, \beta])$*;*
6. $\mathcal{R}[\Sigma_d, \sigma, \exists x\alpha] = \exists x\mathcal{R}[\Sigma_d, \sigma, \alpha]$*;*
7. $\mathcal{R}[\Sigma_d, \sigma, [t]\alpha] = \mathcal{R}[\Sigma_d, \sigma \cdot t, \alpha]$*;*
8. $\mathcal{R}[\Sigma_d, \sigma, Poss(t)] = \mathcal{R}[\Sigma_d, \sigma, \pi^a_t]$*;*
9. $\mathcal{R}[\Sigma_d, \sigma, F(\vec{t})]$ *is defined inductively on* σ *by:*
   (a) $\mathcal{R}[\Sigma_d, \langle\,\rangle, F(\vec{t})] = F(\vec{t})$*;*
   (b) $\mathcal{R}[\Sigma_d, \sigma \cdot t, F(\vec{t})] = \mathcal{R}[\Sigma_d, \sigma, (\gamma_F)^{a\,\vec{x}}_{t\,\vec{t}}]$.

Lemma 3 *For any* α*,* $\mathcal{R}[\Sigma_d, \alpha]$ *is static.*

Semantics

For defining the logic's semantics, we need the following definitions from (Liu, Lakemeyer, and Levesque 2004):

Definition 4 (Unit Propagation) *A* clause *is a disjunction of literals, where a* literal *is either a ground atom* $F(\vec{t})$ *or its negation* $\neg F(\vec{t})$*. In a* unit resolution step*, we infer a clause* c *from a unit clause* $\{l\}$ *and some clause* $\{\bar{l}\} \cup c$*, where* $\bar{l}$ *refers to the complement of literal* l*. Let* s *be a (possibly infinite) set of ground clauses. A* unit derivation *of a clause* c *from* s *is given by a sequence* $c_1, \ldots, c_n$*, where* c_n *is* c *and each* c_i *is either an element from* s *or derivable from previous clauses by unit resolution. We then denote the closure of* s *under unit resolution by* $UR(s)$*, which is the set of clauses* c *such that there is some unit derivation of* c *from* s*. Further,* $US(s)$ *is the set of ground clauses* c *such that* c *is subsumed by some clause in* $UR(s)$*.*

Definition 5 (Belief Reduction)

1. $(\mathbf{B}_k c)\!\downarrow = \mathbf{B}_k c$*, where* c *is a clause;*
2. $(\mathbf{B}_k(t = t'))\!\downarrow = (t = t')$*;*
3. $(\mathbf{B}_k\neg(t = t'))\!\downarrow = \neg(t = t')$*;*
4. $(\mathbf{B}_k\neg\neg\phi)\!\downarrow = \mathbf{B}_k\phi$*;*
5. $(\mathbf{B}_k(\phi \lor \psi))\!\downarrow = (\mathbf{B}_k\phi \lor \mathbf{B}_k\psi)$*, where* $\phi \lor \psi$ *is not a clause;*
6. $(\mathbf{B}_k\neg(\phi \lor \psi))\!\downarrow = (\mathbf{B}_k\neg\phi \land \mathbf{B}_k\neg\psi)$*;*
7. $(\mathbf{B}_k\exists x\phi)\!\downarrow = \exists x\mathbf{B}_k\phi$*;*
8. $(\mathbf{B}_k\neg\exists x\phi)\!\downarrow = \forall x\mathbf{B}_k\neg\phi$*.*

We can now define the semantics of formulas. A semantic model is given by two things: a *setup* s, which is a (possibly infinite) set of nonempty ground clauses, and represents the agent's explicit beliefs about the current world state. Furthermore we need some $\Sigma_d = \Sigma_{\text{pre}} \cup \Sigma_{\text{post}}$, which represents the agent's explicit beliefs about the world's dynamics.

Definition 6 (Semantics of Formulas)

1. $s \models_{\Sigma_d} (d_1 = d_2)$ *iff* d_1 *and* d_2 *are identical object constants;*
2. $s \models_{\Sigma_d} \neg\varphi$ *iff* $s \not\models_{\Sigma_d} \varphi$*;*
3. $s \models_{\Sigma_d} \varphi_1 \lor \varphi_2$ *iff* $s \models_{\Sigma_d} \varphi_1$ *or* $s \models_{\Sigma_d} \varphi_2$*;*
4. $s \models_{\Sigma_d} \exists x\varphi$ *iff* $s \models_{\Sigma_d} \varphi^x_d$ *for some object constant* d*;*
5. $s \models_{\Sigma_d} \mathbf{B}_k\phi$ *iff one of the following holds:*
   (a) subsumption*:*
   $k = 0$*,* ϕ *is a clause* c*, and* $c \in US(s)$*;*
   (b) reduction*:*
   ϕ *is static, but not a clause and* $s \models_{\Sigma_d} (\mathbf{B}_k\phi)\!\downarrow$*;*
   (c) splitting*:*
   $k > 0$*,* ϕ *is static and there is some* $c \in s$ *such that for all* $\rho \in c$*,* $s \cup \{\rho\} \models_{\Sigma_d} \mathbf{B}_{k-1}\phi$*;*
   (d) regression*:*
   ϕ *is not static, and* $s \models_{\Sigma_d} \mathbf{B}_k(\mathcal{R}[\Sigma_d, \phi])$*.*

The notation φ^x_t denotes φ with all free occurrences of x replaced by t. Apart from item 5d and the extra Σ_d argument, the semantical definition is identical to the one in (Liu, Lakemeyer, and Levesque 2004). Item 5a says that anything derivable from s by unit propagation is also available at belief level zero, since unit derivations are computationally cheap. According to item 5b, something is believed at level k when a corresponding simpler formula is already believed. Item 5c encodes that case distinctions make reasoning computationally expensive: ϕ is believed at level k when for some clause we can make a case distinction over all its literals and ϕ holds at level $k-1$ in any case. Finally, our addition of item 5d means that a formula involving actions is believed at level k iff its regression is. Because the regression is a static formula, one of the other three cases needs to be applied subsequently.

A sentence α is valid wrt Σ_d, written $\models_{\Sigma_d} \alpha$, if for every setup s, $s \models_{\Sigma_d} \alpha$. If α does not contain any actions, we often also leave out the Σ_d subscript. Typically, we are interested in checking whether, given a set of explicit belief Σ_0, some ϕ holds at believe level k. We therefore introduce the notation $\Sigma_0 \cup \Sigma_d \models_k \phi$ as an abbreviation for $\models_{\Sigma_d} \mathbf{B}_0\Sigma_0 \supset \mathbf{B}_k\phi$, again possibly leaving out Σ_d when no actions are involved.

Properties

When restricted to static formulas, $\mathcal{SLA}$ is identical to $\mathcal{SL}$:

Theorem 7 *Let ϕ be static. Then*

$$\mathbf{B}_0\Sigma_0 \models_{\Sigma_d} \mathbf{B}_k\phi \;\textit{ iff }\; \models_{\mathcal{SL}} \mathbf{B}_0\Sigma_0 \supset \mathbf{B}_k\phi.$$

Furthermore, we have the following soundness result in terms of entailment of $\mathcal{ES}$ formulas:

Theorem 8 *If $\Sigma_0 \cup \Sigma_d \models_k \phi$, then $\Sigma_0 \cup \Sigma_d \models_{\mathcal{ES}} \phi$.*

This result also establishes the connection to the classical Situation Calculus, of which $\mathcal{ES}$ may be considered a modal dialect. For the details of the two formalisms' relation, we refer the interested reader to (Lakemeyer and Levesque 2005).

We can now reuse results related to these two logics, in particular concerning efficient reasoning with proper$^+$ knowledge bases as defined in (Liu and Levesque 2005):

Definition 9 (Proper$^+$ KBs) *A KB is proper$^+$ if it is a non-empty set of formulas of the form $\forall(e \supset c)$, where e is an ewff and c is a disjunction of literals whose arguments are distinct variables. An* ewff *is a static, quantifier-free formula without fluents and equalities among action terms.*

It is easy to see that the example Σ_0 (1) can be represented in proper$^+$ form. Reasoning with such KBs is tractable in the following sense:

Theorem 10 ((Liu and Levesque 2005)) *If Σ_0 is proper$^+$, ϕ static, and Σ_0 and ϕ use at most j different variables, then whether $\Sigma_0 \models_k \phi$ can be decided in time $O((ln^{j+1})^{k+1})$, where l is the size of ϕ, and n the size of Σ_0.*

That is, reasoning is only exponential in the number of variables used and the belief level. When the ϕ in question is not static, it first needs to be regressed. As the result again is a static formula, the same reasoning procedure can be used. It should however be noted that in the worst case, the length l of the regression result may be in turn exponential in the number of nested occurrences of $[t]$, since in each regression step, a fluent atom is replaced by an entire formula.

Programs

Our program semantics follows the one in (Claßen and Lakemeyer 2008), which is an adaptation of the single step semantics of (De Giacomo, Lespérance, and Levesque 2000). Given a setup s, some Σ_d, a believe level k, and a sequence z of already executed actions, a program δ is mapped to a set of action sequences z', which we call *program execution traces*. The definition uses the notion of program configurations (δ, z), where δ is a program (intuitively what remains to be executed) and z a sequence of ground actions (that have already been performed). A final configuration is one where program execution may legally and successfully terminate, and single step transitions t turn a configuration (δ, z) into a new configuration $(\delta', z \cdot t)$.

Formally, the set of final configurations $\mathcal{F}^{\Sigma_d}_{s,k}$ is the smallest set such that for all δ, δ_1, δ_2, static ϕ and z:

1. $(\phi?, z) \in \mathcal{F}^{\Sigma_d}_{s,k}$ if $s \models_{\Sigma_d} \mathbf{B}_k([z]\phi)$;
2. $(\delta_1; \delta_2, z) \in \mathcal{F}^{\Sigma_d}_{s,k}$ if $(\delta_1, z) \in \mathcal{F}^{\Sigma_d}_{s,k}$ and $(\delta_2, z) \in \mathcal{F}^{\Sigma_d}_{s,k}$;
3. $(\delta_1 | \delta_2, z) \in \mathcal{F}^{\Sigma_d}_{s,k}$ if $(\delta_1, z) \in \mathcal{F}^{\Sigma_d}_{s,k}$ or $(\delta_2, z) \in \mathcal{F}^{\Sigma_d}_{s,k}$;
4. $(\pi x.\delta, z) \in \mathcal{F}^{\Sigma_d}_{s,k}$
   if $(\delta^x_d, z) \in \mathcal{F}^{\Sigma_d}_{s,k}$ for some object constant d;
5. $(\delta^*, z) \in \mathcal{F}^{\Sigma_d}_{s,k}$.

Thus, a configuration $(\phi?, z)$ whose remaining program is a test is final wrt s, Σ_d and k if the formula[1] $[z]\phi$ is believed at level k in s and Σ_d. From the above it also follows that $(t, z) \notin \mathcal{F}^{\Sigma_d}_{s,k}$ for atomic t, i.e. if some action t remains to be done, the configuration cannot be final. Further, sequences are only final when the involved subprograms are both final etc. The transition relation among program configurations is given as follows (the empty program nil abbreviates $\top$?):

1. $(t, z) \underset{s,\Sigma_d,k}{\longrightarrow} (nil, z \cdot t)$;
2. $(\delta_1; \delta_2, z) \underset{s,\Sigma_d,k}{\longrightarrow} (\gamma; \delta_2, z \cdot t)$ if $(\delta_1, z) \underset{s,\Sigma_d,k}{\longrightarrow} (\gamma, z \cdot t)$;
3. $(\delta_1; \delta_2, z) \underset{s,\Sigma_d,k}{\longrightarrow} (\delta', z \cdot t)$
   if $(\delta_1, z) \in \mathcal{F}^{\Sigma_d}_{s,k}$ and $(\delta_2, z) \underset{s,\Sigma_d,k}{\longrightarrow} (\delta', z \cdot t)$;
4. $(\delta_1 | \delta_2, z) \underset{s,\Sigma_d,k}{\longrightarrow} (\delta', z \cdot t)$
   if $(\delta_1, z) \underset{s,\Sigma_d,k}{\longrightarrow} (\delta', z \cdot t)$ or $(\delta_2, z) \underset{s,\Sigma_d,k}{\longrightarrow} (\delta', z \cdot t)$;
5. $(\pi x.\delta, z) \underset{s,\Sigma_d,k}{\longrightarrow} (\delta', z \cdot t)$
   if $(\delta^x_d, z) \underset{s,\Sigma_d,k}{\longrightarrow} (\delta', z \cdot t)$ for some object constant d;
6. $(\delta^*, z) \underset{s,\Sigma_d,k}{\longrightarrow} (\gamma; \delta^*, z \cdot t)$ if $(\delta, z) \underset{s,\Sigma_d,k}{\longrightarrow} (\gamma, z \cdot t)$.

If $\underset{s,\Sigma_d,k}{\overset{*}{\longrightarrow}}$ is the reflexive transitive closure of $\underset{s,\Sigma_d,k}{\longrightarrow}$, then

$$\{z' \mid (\delta, z) \underset{s,\Sigma_d,k}{\overset{*}{\longrightarrow}} (\delta', z \cdot z') \text{ and } (\delta', z \cdot z') \in \mathcal{F}^{\Sigma_d}_{s,k}\}$$

is the set $\|\delta\|^{s,k}_{\Sigma_d}(z)$ of execution traces of δ, given s, k, Σ_d, at z. Such a trace therefore corresponds to a (possibly empty) sequence of transition steps that lead into a final configuration. Note that because of rule 1, the actions contained in the

[1] We extend $[\cdot]$ to sequences: $[\langle\rangle]\phi \overset{def}{=} \phi$, $[z \cdot t]\phi \overset{def}{=} [z][t]\phi$.

trace are not necessarily all executable according to *Poss*, to allow for reasoning about hypothetical situations including non-reachable ones. In case we are only interested in the actually executable traces of our program δ, we simply have to substitute each occurrence of an action t by $Poss(t)?; t$ in δ.

Given an initial KB Σ_0, we further define

$$\|\delta\|^{\Sigma_0,k}_{\Sigma_d}(z) \overset{def}{=} \bigcap \{\|\delta\|^{s,k}_{\Sigma_d}(z) \mid s \models \mathbf{B}_0\Sigma_0\}$$

to be the set of execution traces common to all setups s where Σ_0 is explicitly believed. Our program semantics is sound (but not complete) wrt the program semantics of $\mathcal{ESG}$ as presented in (Claßen and Lakemeyer 2008) as follows:

Theorem 11 *If $z' \in \|\delta\|^{\Sigma_0,k}_{\Sigma_d}(z)$, then for any semantic model w of $\mathcal{ESG}$ such that $w \models \Sigma_0 \cup \Sigma_d$, also $z' \in \|\delta\|^{w}(z)$.*

Again the relation to classical Golog is given by the results presented in (Claßen and Lakemeyer 2008) and (Lakemeyer and Levesque 2005).

Execution of Programs

In classical Golog, programs are executed off-line, meaning the interpreter first analyzes the entire program to search for a conforming execution trace before performing any actions in the real world. This soon becomes infeasible, in particular when the program is large, the agent has only incomplete world knowledge and has to use sensing to gather information at run-time. IndiGolog (Sardina et al. 2004) therefore executes programs on-line, which means that there is no general look-ahead, but the system just does the next possible action in each step, treating nondeterminism like random choices. Look-ahead is only applied to parts of the program that are explicitly marked by the search operator $\Sigma(\delta)$, thus giving the programmer the control over where the system should spend computational effort for searching. However the search does not pay attention to the computational costs of plans. The main contribution of this paper is to propose the following two new offline search operators:

- $\Lambda_k(\delta, z)$,
  where the set of solution traces is $\|\delta\|^{\Sigma_0,k}_{\Sigma_d}(z)$;
- $\Lambda_*(\delta, z)$,
  where the set of solution traces $\bigcup_{k=0}^{\infty} \|\delta\|^{\Sigma_0,k}_{\Sigma_d}(z)$.

Whereas Λ_k only finds solutions obtainable by reasoning up to belief level k, Λ_* considers all belief levels, where the idea is that lower level solutions will be tested before ones at higher levels, thus preferring plans that require the least computational costs.

The algorithms we are going to present here for computing according solutions make use of the notion of characteristic program graphs as presented in (Claßen and Lakemeyer 2008). Due to space reasons, we will not repeat the (entire) definition here. Intuitively, a program δ is mapped to a graph $\mathcal{G}_\delta = \langle V, E, v_0 \rangle$, with[2]

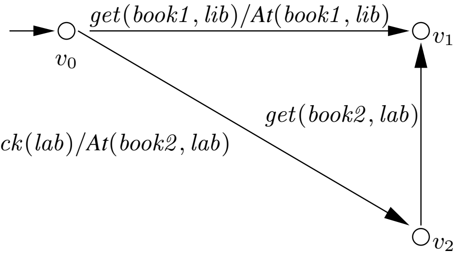
$get(book1, lib)/At(book1, lib)$
v_0
v_1
$get(book2, lab)$
$unlock(lab)/At(book2, lab)$
v_2

Figure 1: Characteristic Graph of Example Program

- V is a set of vertices of the form $\langle \delta', \varphi' \rangle$, where the δ' is some remaining subprogram and φ' is an objective formula encoding a condition under which program execution might terminate at that node.
- E is a set of labelled edges of the form $v' \overset{t/\phi}{\to} v''$, where the intuition is that a transition with ground action t may be taken from node v' to node v'' when the objective formula ϕ holds.
- $v_0 = \langle \delta, \varphi_0 \rangle \in V$ is the initial node.

The graph for the example program δ from the introduction is shown in Figure 1. The nodes are $v_0 = \langle \delta, \bot \rangle$, $v_1 = \langle nil, \top \rangle$, and $v_2 = \langle get(book2, lab), \bot \rangle$, where $\top$ denotes truth (definable as $\forall x(x = x)$) and $\bot$ falsity ($\neg\top$).

Definition 12 *Let ξ be a path in the characteristic graph. The* path formula $PF(\xi)$ *is defined inductively on its length:*

- $PF(v) = \varphi'$, *if* $v = \langle \delta', \varphi' \rangle$*;*
- $PF(\xi) = \psi \wedge [t]PF(\xi')$, *if* $\xi = v \overset{t/\psi}{\to} \xi'$.

Further, the path trace $PT(\xi)$ *is defined as*

- $PT(v) = \langle \, \rangle$*;*
- $PT(\xi) = t \cdot PT(\xi')$, *if* $\xi = v \overset{t/\psi}{\to} \xi'$.

For a fixed believe level k, our method now tests paths of increasing lengths.

Procedure 1 Comp$\Lambda_k(\delta, z)$

```
Determine $\mathcal{G}_\delta = \langle V, E, v_0 \rangle$
for $l = 0, 1, 2 \ldots$ do
    for all paths $\xi$ of length $l$ starting in $v_0$ do
        if $\Sigma_0 \cup \Sigma_d \models_k [z]PF(\xi)$ then
            return $PT(\xi)$
```

When the set of possible paths is finite like in our example, Comp$\Lambda_k(\delta, z)$ will always terminate for any k and z. In this case we can call that procedure for increasing belief level k:

Procedure 2 Comp$\Lambda_*(\delta, z)$

```
for $k = 0, \ldots, \infty$ do
    Comp$\Lambda_k(\delta, z)$
```

Let us apply Comp$\Lambda_*(\delta, \langle \, \rangle)$ to our example program δ, and let us assume that no actions have been performed so far by the agent, i.e. $z = \langle \, \rangle$. We first call Comp$\Lambda_0(\delta, \langle \, \rangle)$ for belief level $k = 0$. The program graph contains four different paths

[2] Here we assume that the program in question does not contain any π operators, which keeps things much simpler. In the future work section we discuss how to extend our approach to this case.

that start in the initial node: the only path of length zero, $\xi_0 = v_0$, further two paths of length one, $\xi_{11} = v_0 \rightarrow v_1$ and $\xi_{12} = v_0 \rightarrow v_2$, and finally one path of length two, namely $\xi_2 = v_0 \rightarrow v_2 \rightarrow v_1$. Their respective path formulas are:

$$\begin{array}{lcl} PF(\xi_0) & = & \bot \\ PF(\xi_{11}) & = & \mathit{At}(\mathit{book1}, \mathit{lib}) \wedge [\mathit{get}(\mathit{book1}, \mathit{lib})]\top \\ PF(\xi_{12}) & = & \mathit{At}(\mathit{book2}, \mathit{lab}) \wedge [\mathit{unlock}(\mathit{lab})]\bot \\ PF(\xi_2) & = & \mathit{At}(\mathit{book2}, \mathit{lab}) \wedge \\ & & \quad [\mathit{unlock}(\mathit{lab})](\top \wedge [\mathit{get}(\mathit{book2}, \mathit{lab})]\top) \end{array}$$

Then we need to check for each ξ whether $\Sigma_0 \cup \Sigma_d \models_0 PF(\xi)$, which is the same as $\models_{\Sigma_d} \mathbf{B}_0\Sigma_0 \supset \mathbf{B}_0 PF(\xi)$, which according to rule 5d of the semantics means $\models_{\Sigma_d} \mathbf{B}_0\Sigma_0 \supset \mathbf{B}_0(\mathcal{R}[\Sigma_d, PF(\xi)])$. The regressed versions of the path formulas are, with simplifications:

$$\begin{array}{ll} \mathcal{R}[\Sigma_d, PF(\xi_0)] \;= \bot & \mathcal{R}[\Sigma_d, PF(\xi_{11})] = \mathit{At}(\mathit{book1}, \mathit{lib}) \\ \mathcal{R}[\Sigma_d, PF(\xi_{12})] = \bot & \mathcal{R}[\Sigma_d, PF(\xi_2)] \;= \mathit{At}(\mathit{book2}, \mathit{lab}) \end{array}$$

To see that both $\not\models_{\Sigma_d} \mathbf{B}_0\Sigma_0 \supset \mathbf{B}_0\mathit{At}(\mathit{book1}, \mathit{lib})$ as well as $\not\models_{\Sigma_d} \mathbf{B}_0\Sigma_0 \supset \mathbf{B}_0\bot$, let s be the setup given by

$$\begin{array}{l} \{\mathit{At}(\mathit{book2}, \mathit{lab}), \\ \mathit{Borrowed}(\mathit{ann}, \mathit{book1}) \vee \mathit{Borrowed}(\mathit{bob}, \mathit{book1}), \\ \neg\mathit{Borrowed}(d, \mathit{book1}) \vee \mathit{At}(\mathit{book1}, \mathit{lib}) | d \text{ an obj. const.}\}. \end{array}$$

Then $s \models \mathbf{B}_0\Sigma_0$. As both $\bot$ and $\mathit{At}(\mathit{book1}, \mathit{lib})$ are clauses ($\bot$ is the empty clause), the only possibility is that they are believed by subsumption. However, in this case $UR(s) = s$ (no unit propagation is possible) and there is no clause in s that subsumes $\bot$ or $\mathit{At}(\mathit{book1}, \mathit{lib})$, hence $s \not\models \mathbf{B}_0\bot$ and $s \not\models \mathbf{B}_0\mathit{At}(\mathit{book1}, \mathit{lib})$. On the other hand, when s is some arbitrary setup with $s \models \mathbf{B}_0\Sigma_0$, then $s \models \mathbf{B}_0\mathit{At}(\mathit{book2}, \mathit{lab})$ by reduction (treating a set as a conjunction), therefore $\models_{\Sigma_d} \mathbf{B}_0\Sigma_0 \supset \mathbf{B}_0\mathit{At}(\mathit{book2}, \mathit{lab})$. $\textsc{Comp}\Lambda_0(\delta, \langle\rangle)$ thus returns

$$PT(\xi_2) = \langle \mathit{unlock}(\mathit{lab}), \mathit{get}(\mathit{book2}, \mathit{lab})\rangle.$$

When $k = 1$, we get $\models_{\Sigma_d} \mathbf{B}_0\Sigma_0 \supset \mathbf{B}_1\mathit{At}(\mathit{book1}, \mathit{lib})$ as follows. Let again s be a setup with $s \models \mathbf{B}_0\Sigma_0$. Then s will contain a clause that subsumes $\mathit{Borrowed}(\mathit{ann}, \mathit{book1}) \vee \mathit{Borrowed}(\mathit{bob}, \mathit{book2})$, and we can split over this clause. As s also must contain a clause subsuming $\neg\mathit{Borrowed}(\mathit{ann}, \mathit{book1}) \vee \mathit{At}(\mathit{book1}, \mathit{lib})$, $\mathit{At}(\mathit{book1}, \mathit{lib})$ can be obtained from $s \cup \{\mathit{Borrowed}(\mathit{ann}, \mathit{book1})\}$ by unit propagation. Similarly for $\mathit{Borrowed}(\mathit{bob}, \mathit{book1})$, therefore $s \models \mathbf{B}_1\mathit{At}(\mathit{book1}, \mathit{lib})$. Only the $k = 1$ cycle of $\textsc{Comp}\Lambda_*(\delta, \langle\rangle)$ will hence yield the solution

$$PT(\xi_{11}) = \langle \mathit{get}(\mathit{book1}, \mathit{lib})\rangle.$$

Future Work

The approach presented here is work in progress. We are currently working on implementing the method by integrating a corresponding reasoner and search operator into the IndiGolog agent framework (Sardina et al. 2004) to be able to also evaluate it empirically against existing techniques.

There are furthermore many directions for future work at the conceptual level. Instead of solely using regression-based reasoning, we might extend our approach using recent tractability results for the progression of proper$^+$ knowledge bases (Liu and Lakemeyer 2009). As the naive loop of Λ_k obviously will not terminate once the program δ contains an iteration, Λ_* will in this case get stuck at belief level zero, even if there are possibly solutions at higher levels. One may try to apply some sort of dove-tailing technique here, where for any k, only solutions up to a length $l(k)$ are considered.

It is further conceivable to combine our language with an appropriate model for action time costs to be able to study trade-offs between the required reasoning and actual execution time of plans. Also, a variant of our method that computes conditional plans may be useful. In particular, it might be necessary to adapt the notion of *epistemic feasibility* as discussed in (Sardina et al. 2004): Consider a conditional plan in the form of the following program:

$$\phi?; a \;|\; \neg\phi?; b$$

where at plan time, the truth value of ϕ was unknown. To be able to execute this program we have to ensure that ϕ will become known at run time, or the executor gets stuck not knowing what step to take next. We therefore also need to extend our formalism appropriately to allow for *sensing actions* that the agent can use in order to gather the necessary information at run time, possibly in combination with knowledge-based programs as described in (Claßen and Lakemeyer 2006).

Finally, integrating the π operators we omitted in the previous section is straightforward in principle, but somewhat tedious. The idea is that whenever some πx is encountered on a path, the corresponding x in the path formula is substituted by a fresh variable x' as different quantifiers may use identical variable names. The obtained path formula then contains a number of free variables. The tractable reasoning procedure presented in (Liu and Levesque 2005) is able to deal with open queries for which it computes a set of variable substitutions. Each such substitution, applied to a path trace with free variables, then corresponds to one possible solution trace.

Conclusion

In this paper we introduced a new logic called $\mathcal{SLA}$ for tractable reasoning with limited beliefs in the presence of action and change. Based on this, we proposed a new planning operator that considers increasing levels of belief, thus preferring solution plans that are the computationally cheapest to discover.

Acknowledgements

This work was supported by the German National Science Foundation (DFG) under grant La 747/14-1. We also thank the anonymous reviewers for their helpful comments.

References

Claßen, J., and Lakemeyer, G. 2006. Foundations for knowledge-based programs using ES. In Doherty, P.; Mylopoulos, J.; and Welty, C. A., eds., *KR*, 318–318. AAAI Press.

Claßen, J., and Lakemeyer, G. 2008. A logic for non-terminating Golog programs. In Brewka, G., and Lang, J., eds., *KR*, 589–599. AAAI Press.

De Giacomo, G.; Lespérance, Y.; and Levesque, H. 2000. ConGolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence* 121(1–2):109–169.

Ferrein, A., and Lakemeyer, G. 2008. Logic-based robot control in highly dynamic domains. *Robot. Auton. Syst.* 56(11):980–991.

Lakemeyer, G., and Levesque, H. J. 2004. Situations, si! situation terms, no! In *KR*, 516–526. AAAI Press.

Lakemeyer, G., and Levesque, H. J. 2005. Semantics for a useful fragment of the situation calculus. In Kaelbling, L. P., and Saffiotti, A., eds., *IJCAI*, 490–496. Professional Book Center.

Levesque, H.; Reiter, R.; Lespérance, Y.; Lin, F.; and Scherl, R. B. 1997. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming* 31:59–84.

Liu, Y., and Lakemeyer, G. 2009. On first-order definability and computability of progression for local-effect actions and beyond. In *IJCAI*.

Liu, Y., and Levesque, H. J. 2005. Tractable reasoning in first-order knowledge bases with disjunctive information. In *AAAI*, 639–644. AAAI Press.

Liu, Y.; Lakemeyer, G.; and Levesque, H. 2004. A logic of limited belief for reasoning with disjunctive information. In *KR*, 587–597.

McCarthy, J., and Hayes, P. 1969. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence 4*. New York: American Elsevier. 463–502.

Reiter, R. 2001. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press.

Sardina, S.; De Giacomo, G.; Lespérance, Y.; and Levesque, H. J. 2004. On the semantics of deliberation in Indigolog—from theory to implementation. *Annals of Mathematics and Artificial Intelligence* 41(2-4):259–299.

A Psychological Study of Comparative Non-monotonic Preferences Semantics

Rui da Silva Neves
Université Toulouse-II,
CLLE-LTC, CNRS UMR 5263
5 Allées Machado
31058 Toulouse Cedex 9, France
neves@univ-tlse2.fr

Souhila Kaci
Université Lille-Nord de France, Artois
CRIL, CNRS UMR 8188
IUT de Lens
F-62307, France
kaci@cril.fr

Abstract

Representing preferences and reasoning about them are important issues for many real-life applications. Several monotonic and non-monotonic qualitative formalisms have been developed for this purpose. Most of them are based on comparative preferences, for e.g. "I prefer red wine to white wine". However this simple and natural way to express preferences comes also with many difficulties regarding their interpretation. Several (more or less strong) semantics have been proposed leading to different (pre)orders on outcomes. In this paper, we report results of the first empirical comparison of existing non-monotonic semantics (strong, optimistic, pessimistic and ceteris paribus) based on psychological data. Thirty participants were asked to rank 8 menus according to their preferences and to compare 31 pairs of menus. The recorded preferences allowed to compute compact preferences and ranking menus for each participant according to the four semantics under study, and to compare these ranks to participant's ones. Results show that non-monotonic optimistic and pessimistic preferences are the semantics that better fit human data, strong and ceteris paribus semantics being less psychologically plausible given our task.

Introduction

Preferences are very useful in many real-life problems. They are inherently a multi-disciplinary topic, of interest to economists, computer scientists, operations researchers, mathematicians, logicians, philosophers and psychologists.

It has been early recognized that value functions/orderings cannot be explicitly defined because of a great number of outcomes or simply because the user is not willing to state her/his preferences on each pair of outcomes. Indeed preferences should be handled in a compact (or succint) way, starting from non completely explicit preferences expressed by a user.

The compact languages for preference representation have been extensively developed in Artificial Intelligence in the last decade (Boutilier et al. 2004; Brewka, Benferhat, and Le Berre 2004). In particular, (conditional) comparative statements are often used for describing preferences in a local, contextualized manner for e.g., "I prefer fish to meat", "if meat is served then I prefer red wine to white wine", etc. Indeed, it is easier and more natural to express such qualitative comparative statements than to say that I prefer fish with the weight .8 and prefer meat with the weight .2. Some generic principles are often used for completing the qualitative comparative preference statements[1] (Hansson 1996; Boutilier 1994; Benferhat et al. 2002). Although comparative preference statements allow for a simple and natural way to express preferences, they come however with many difficulties regarding their interpretation.

Comparative preferences are often interpreted following the well known ceteris paribus semantics (Hansson 1996). This is due to the CP-net approach (Boutilier et al. 2004) which has emerged in the last decade as the preeminent and prominent method for processing preferences in Artificial Intelligence, thanks to its intuitive appeal. Following this pinciple, the statement "I prefer fish to meat" is interpreted as, given two meals that differ *only* in the main dish, the meal with fish is preferred to the meal with meat. However, CP-nets behave monotonically and do not allow for the handling of preferences with defaults. For example, we can prefer fish to meat, but when available fish is red tuna and meat is poultry, we can prefer the reverse. Moreover, in CP-nets, ceteris paribus semantics states that the two meals *fish-cake* and *meat-ice cream* are incomparable w.r.t. the preference statement "I prefer fish to meat" while a vegetarian would prefer any fish-based meal to any meat-based meal. Fortunately, ceteris paribus is not the only possible reading of comparative preference statements and other intuitively non-monotonic meaningful semantics may also be encountered, and researchers have also argued for other semantics (Boutilier 1994; Benferhat et al. 2002) based on insights from non-monotonic reasoning such as system Z (Pearl 1990). Note also that ceteris paribus semantics can also be non-monotonic outside CP-net framework. For example, the menu $fish - red$ is preferred to the menu $fish - white$ w.r.t. the preference statement "red is preferred to $\neg red$" following ceteris paribus semantics. However the additional preference statement "$fish \wedge white$ is preferred to $fish \wedge \neg white$" induces the reverse preference, namely $fish - white$ is preferred to $fish - red$.

In this paper, we provide the first empirical compar-

[1] From now on, we simply speak about comparative preference statements.

ison of existing non-monotonic semantics (including ceteris paribus) based on psychological data. This psychological inquiry is founded by previous work on the non-monotonic nature of human reasoning. For example, it has been shown that human inference is consistent with System P (Kraus, Lehmann, and Magidor 1990) (see (Neves, Bonnefon, and Raufaste 2002; Benferhat, Bonnefon, and Da Silva Neves 2004)) and that System P constitutes a psychologically sound base of rationality postulates for the evaluation of non-monotonic reasoning systems. In our study, participants were asked to rank 8 menus according to their preferences and to compare 31 pairs of menus. The recorded preferences were compared to those provided by the considered semantics. Results show that optimistic and pessimistic preferences are the semantics that better fit human data, strong, ceteris paribus semantics being less psychologically plausible given our task.

The remainder of this paper is organized as follows. After providing notations and necessary definitions, we recall the different semantics of comparatives preferences proposed in literature. Then, we recall algorithms to rank-order outcomes for each semantics. In the next section, we provide empirical comparison of the different semantics based on psychological data. Lastly we conclude.

Notations

Let $V = \{X_1, \cdots, X_h\}$ be a set of h variables. Each variable X_i takes its values in a domain $Dom(X_i)$ which is a set of uninterpreted constants $\mathcal{D}$ or rational numbers $\mathcal{Q}$. A possible outcome, denoted t, is the result of assigning a value in $Dom(X_i)$ to each variable X_i in V. Ω is the set of all possible outcomes. We suppose that this set is fixed and finite. Let $\mathcal{L}$ be a language based on V. $Mod(\varphi)$ denotes the set of outcomes that make the formula φ (built on $\mathcal{L}$) true. We write $t \models \varphi$ when $t \in Mod(\varphi)$ and say that t satisfies φ.

An ordering relation $\succeq$ on $\mathcal{X} = \{x, y, z, \cdots\}$ is a reflexive binary relation such that $x \succeq y$ stands for x is at least as preferred as y. $x \approx y$ means that both $x \succeq y$ and $y \succeq x$ hold, i.e., x and y are equally preferred. Lastly $x \sim y$ means that neither $x \succeq y$ nor $y \succeq x$ holds, i.e., x and y are incomparable. A strict ordering relation on $\mathcal{X}$ is an irreflexive binary relation such that $x \succ y$ means that x is strictly preferred to y. We also say that x dominates y. A strict ordering relation $\succ$ can be defined from an ordering relation $\succeq$ as $x \succ y$ if $x \succeq y$ holds but $y \succeq x$ does not.

When neither $x \succ y$ nor $y \succ x$ holds, we also write $x \sim y$. $\succeq$ (resp. $\succ$) is a preorder (resp. order) on $\mathcal{X}$ if and only if $\succeq$ (resp. $\succ$) is transitive, i.e., if $x \succeq y$ and $y \succeq z$ then $x \succeq z$ (if $x \succ y$ and $y \succ z$ then $x \succ z$). $\succeq$ (resp. $\succ$) is a complete preorder (resp. order) if and only if $\forall x, y \in \mathcal{X}$, we have either $x \succeq y$ or $y \succeq x$ (resp. either $x \succ y$ or $y \succ x$).

The set of the best (or undominated) elements of $A \subseteq \mathcal{X}$ w.r.t. $\succ$, denoted $\max(A, \succ)$, is defined by $\max(A, \succ) = \{x | x \in A, \nexists y \in A, y \succ x\}$. The set of the worst elements of $A \subseteq \mathcal{X}$ w.r.t. $\succ$, denoted $\min(A, \succ \ )$, is defined by $\min(A, \succ \ ) = \{x | x \in A, \nexists y \in A, x \succ y\}$. The best (resp. worst) elements of A w.r.t. $\succeq$ is $\max(A, \succ)$ (resp. $\min(A, \succ)$) where $\succ$ is the strict ordering relation associated to $\succeq$.

A complete preorder $\succeq$ can also be represented by a well ordered partition of Ω. This is an equivalent representation, in the sense that each preorder corresponds to one ordered partition and vice versa.

Definition 1 (Partition) *A sequence of sets of outcomes of the form $(E_1, \ldots, E_n)$ is a partition of Ω if and only if (i) $\forall i$, $E_i \neq \emptyset$, (ii) $E_1 \cup \cdots \cup E_n = \Omega$, and (iii) $\forall i, j$, $E_i \cap E_j = \emptyset$ for $i \neq j$.*

A partition of Ω is ordered if and only if it is associated with a preorder $\succeq$ on Ω such that ($\forall t, t' \in \Omega$ with $t \in E_i, t' \in E_j$ we have $i \leq j$ if and only if $t \succeq t'$).

Comparative preference statements

We denote comparative statements of the form "I prefer p to q" as $p > q$ and denote conditional (called also contextual) comparative statements of the form "if r is true then I prefer p to q" as $r : p > q$, where p, q and r are any propositional formulas.

Comparative statements come with difficulties regarding their interpretation. How should we interpret such statements? For example, given the preference statement "I prefer fish to meat", how do we rank-order meals based on fish and those based on meat? Four semantics have been proposed in literature:

- *ceteris paribus preferences:* (Hansson 1996)
  any fish-based meal is preferred to any meat-based meal if the two meals are exactly the same elsewhere (for example wine and dessert).
- *strong preferences:* (Boutilier 1994)
  any fish-based meal is preferred to any meat-based meal.
- *optimistic preferences:* (Benferhat, Dubois, and Prade 1992; Boutilier 1994; Pearl 1990)
  at least one fish-based meal is preferred to all meat-based meals.
- *pessimistic preferences:* (Benferhat et al. 2002)
  at least one meat-based meal is less preferred to all fish-based meals.

We define preference of the formula p over the formula q as preference of $p \wedge \neg q$ over $\neg p \wedge q$. This is standard and known as von Wright's expansion principle (von Wright 1963). Additional clauses may be added for the cases in which sets of outcomes are nonempty, to prevent the satisfiability of preferences like $p > \top$ and $p > \bot$. We do not consider this borderline condition to keep the formal machinery as simple as possible. We denote the preference of p over q following strong semantics (resp. ceteris paribus, optimistic, pessimistic) by $p >_{st} q$ (resp. $p >_{cp} q$, $p >_{opt} q$, $p >_{pes} q$).

Definition 2 *Let p and q be two propositional formulas and $\succeq$ be a preorder on Ω.*

- *$\succeq$ satisfies $p >_{st} q$, denoted $\succeq \models p >_{st} q$, iff $\forall t \models p \wedge \neg q$, $\forall t' \models \neg p \wedge q$ we have $t \succ t'$.*
- *$\succeq$ satisfies $p >_{cp} q$, denoted $\succeq \models p >_{cp} q$, iff $\forall t \models p \wedge \neg q$, $\forall t' \models \neg p \wedge q$ we have $t \succ t'$, where t and t' have the same assignment on variables that do not appear in p and q.*

- $\succeq$ *satisfies* $p >_{opt} q$, *denoted* $\succeq \models p >_{opt} q$, *iff* $\exists t \models p \wedge \neg q, \forall t' \models \neg p \wedge q$ *we have* $t \succ t'$.
- $\succeq$ *satisfies* $p >_{pes} q$, *denoted* $\succeq \models p >_{pes} q$, *iff* $\exists t' \models \neg p \wedge q, \forall t \models p \wedge \neg q$ *we have* $t \succ t'$.

A preference set is a set of preferences of the same type.

Definition 3 (Preference set) *A preference set of type* $\rhd$, *denoted* $\mathcal{P}_{\rhd}$, *is a set of preferences of the form* $\{p_i \rhd q_i | i = 1, \cdots, n\}$, *where* $\rhd \in \{ >_{st}, >_{cp}, >_{opt}, >_{pes} \}$. *A complete preorder* $\succeq$ *is a model of* $\mathcal{P}_{\rhd}$ *if and only if* $\succeq$ *satisfies each preference* $p_i \rhd q_i$ *in* $\mathcal{P}_{\rhd}$.

A set $\mathcal{P}_{\rhd}$ is consistent if it has a model.

From comparative preference statements to preorders on outcomes

Generally we have to deal with several comparative preference statements expressed by a user. Once the semantics is fixed, the problem to tackle is how to deal with such statements? Several types of queries can be asked about preferences: what are the preferred outcomes? Is one outcome better than the other? In many applications (for e.g. database queries), users are more concerned with the preferred outcomes. However preferred outcomes are not always feasible. For example the best menus w.r.t. a user's preferences may be no longer available so we have to look for menus that are immediately less preferred w.r.t. user's preferences. In such a case a complete preorders on menus is needed to answer user's preferences. Indeed we restrict ourselves to semantic models that derive complete preorders on outcomes. In the following, we recall algorithms which derive a unique complete preorder given a set of preferences of the same type w.r.t. specificity principle (Yager 1983).

Let $\mathcal{P}_{\rhd} = \{s_i : p_i \rhd q_i | i = 1, \cdots, n\}$ be a preference set with $\rhd \in \{ >_{st}, >_{cp}, >_{opt}, >_{pes} \}$. Given $\mathcal{P}_{\rhd}$, we define a set of pairs on Ω as follows:

$$\mathcal{L}(\mathcal{P}_{\rhd}) = \{C_i = (L(s_i), R(s_i)) | i = 1, \cdots, n\},$$

where $L(s_i) = \{t | t \in \Omega, t \models p_i \wedge \neg q_i\}$ and $R(s_i) = \{t | t \in \Omega, t \models \neg p_i \wedge q_i\}$.

Example 1 *Let* dish, wine *and* dessert *be three variables such that* $Dom(dish) = \{fish, meat\}$, $Dom(wine) = \{white, red\}$ *and* $Dom(dessert) = \{cake, ice\_cream\}$. *We have* $\Omega = \{t_0 = fish - white - ice\_cream,$ $t_1 = fish - white - cake, t_2 = fish - red - ice\_cream,$ $t_3 = fish - red - cake, t_4 = meat - white - ice\_cream,$ $t_5 = meat - white - cake, t_6 = meat - red - ice\_cream,$ $t_7 = meat - red - cake\}$.
Let $\mathcal{P}_{\rhd} = \{s_1 : fish \rhd meat, s_2 : red \wedge cake \rhd white \wedge ice\_cream, s_3 : fish \wedge white \rhd fish \wedge red\}$. *We have* $\mathcal{L}(\mathcal{P}_{\rhd}) = \{C_1 = (\{t_0, t_1, t_2, t_3\}, \{t_4, t_5, t_6, t_7\}),$ $C_2 = (\{t_3, t_7\}, \{t_0, t_4\}), C_3 = (\{t_0, t_1\}, \{t_2, t_3\})\}$.

Optimistic preferences

Several complete preorders may satisfy a set of optimistic preferences. It is however possible to characterize a unique preorder among them under certain assumption. The semantics of optimistic preferences is close to the one of conditionals. Indeed system Z (Pearl 1990) has been used (Benferhat, Dubois, and Prade 1992; Boutilier 1994). It rank-orders outcomes under the assumption that outcomes are preferred unless the contrary is stated. Indeed outcomes are put in the highest possible rank in the preorder while being consistent with preferences at hand. This principle ensures that the complete preorder is unique and the most compact one among preorders satisfying the set of preferences[2]. Algorithm 1 gives the way this preorder is computed. At each step of the algorithm, we put in E_i outcomes that are not dominated by any other outcomes. These outcomes are those which do not appear in the right-hand side of any pair $(L(s_i), R(s_i))$ of $\mathcal{L}(\mathcal{P}_{>_{opt}})$.

Algorithm 1: A complete preorder associated with $\mathcal{P}_{>_{opt}}$.

```
Data: A preference set $\mathcal{P}_{>_{opt}}$.
Result: A complete preorder $\succeq$ on $\Omega$.
begin
    $l = 0$
    while $\Omega \neq \emptyset$ do
        $l = l + 1$
        $E_l = \{t | t \in \Omega, \nexists (L(s_i), R(s_i)) \in \mathcal{L}(\mathcal{P}_{>_{opt}}), t \in R(s_i)\}$
        if $E_l = \emptyset$ then
            stop (inconsistent preferences), $l = l - 1$
        - $\Omega = \Omega \backslash E_l$
        /** remove satisfied preferences **/
        - remove $(L(s_i), R(s_i))$ where $L(s_i) \cap E_l \neq \emptyset$
    return $\succeq = (E_1, \cdots, E_l)$.
end
```

Example 2 *(Example 1 con'd) We have* $E_1 = \{t_1\}$. *We remove* C_1 *and* C_3 *since* $s_1 = fish >_{opt} meat$ *and* $s_3 : fish \wedge white >_{opt} fish \wedge red$ *are satisfied. We get* $\mathcal{L}(\mathcal{P}_{>_{opt}}) = \{C_2 = (\{t_3, t_7\}, \{t_0, t_4\})\}$. *Now* $E_2 = \{t_2, t_3, t_5, t_6, t_7\}$. *We remove* C_2 *since* $s_2 : red \wedge cake >_{opt} white \wedge ice\_cream$ *is satisfied. So* $\mathcal{L}(\mathcal{P}_{>_{opt}}) = \emptyset$. *Lastly,* $E_3 = \{t_0, t_4\}$. *Indeed* $\succeq = (\{t_1\}, \{t_2, t_3, t_5, t_6, t_7\}, \{t_0, t_4\})$. *We can check that each outcome has been put in the highest possible rank in* $\succeq$. *Therefore, if we push an outcome to a higher rank then the preorder does not satisfy the preference set. For example,* $\succeq' = (\{t_1, t_5\}, \{t_2, t_3, t_6, t_7\}, \{t_0, t_4\})$ *does not satisfy* $s_1 = fish >_{opt} meat$.

Pessimistic preferences

The converse reasoning is drawn when dealing with pessimistic preferences (Benferhat et al. 2002). The basic principle is that outcomes are not preferred unless the contrary is stated. Indeed outcomes are put in the lowest possible rank in the preorder while being consistent with preferences at

[2] Technically speaking, this preorder can be obtained by max-based aggregation operator of all preorders satisfying the set of preferences

hand. This principle also ensures that the complete preorder is unique and the most compact one among preorders satisfying the set of preferences[3]. Algorithm 2 gives the way this preorder is computed.

```
Algorithm 2: A complete preorder associated with $\mathcal{P}_{>_{pes}}$ .
Data: A preference set $\mathcal{P}_{>_{pes}}$ .
Result: A complete preorder $\succeq$ on $\Omega$.
begin
    $l = 0$
    while $\Omega \neq \emptyset$ do
        $l = l + 1$
        $E_l = \{t | t \in \Omega, \nexists (L(s_i), R(s_i)) \in \mathcal{L}(\mathcal{P}_{>_{pes}}), t \in L(s_i)\}$
        if $E_l = \emptyset$ then
            stop (inconsistent preferences), $l = l - 1$
        - $\Omega = \Omega \backslash E_l$
        /** remove satisfied preferences **/
        - remove $(L(s_i), R(s_i))$ where $R(s_i) \cap E_l \neq \emptyset$
    return $\succeq = (E'_1, \cdots, E'_l)$ s.t. $0 \leq h \leq l$, $E'_h = E_{l-h+1}$
end
```

Example 3 *(Example 1 con'd) We have $E_1 = \{t_4, t_5, t_6\}$. We remove C_1 and C_2 since $s_1 : fish >_{pes} meat$ and $s_2 : red \wedge cake >_{pes} white \wedge ice\_cream$ are satisfied. We repeat the same reasoning and get $E_2 = \{t_2, t_3, t_7\}$ and $E_3 = \{t_0, t_1\}$. So $\succeq = (\{t_0, t_1\}, \{t_2, t_3, t_7\}, \{t_4, t_5, t_6\})$. We can check that each outcome has been put in the lowest possible rank in the preorder.*

Strong preferences

Strong preferences induce a unique partial order on outcomes. We can use both construction principles used in optimistic and pessimistic preferences to linearize the partial order and compute a unique complete preorder. Algorithms 1 and 2 can be adapted to deal with strong preferences. Due to the lack of space, we only give the algorithm adapting Algorithm 1.

Example 4 *(Example 1 con'd) There is no complete preorder which satisfies $\mathcal{P}_{>_{st}}$ so $\mathcal{P}_{>_{st}}$ is inconsistent. This is due to s_1 and s_2. Following s_1, t_0 is preferred to t_7 while t_7 is preferred to t_0 following s_2.*

Example 5 *(Consistent strong preferences) Let $\mathcal{P}_{>_{st}} = \{fish \wedge white >_{st} fish \wedge red, red \wedge cake >_{st} red \wedge ice\_cream, meat \wedge red >_{st} meat \wedge white\}$. Then following Algorithm 3, we have $\succeq = (\{t_0, t_1, t_7\}, \{t_3\}, \{t_2, t_6\}, \{t_4, t_5\})$. Now following the adaptation of Algorithm 2 to deal with strong preferences, we have $\succeq = (\{t_0, t_1\}, \{t_3, t_7\}, \{t_6\}, \{t_2, t_4, t_5\})$.*

Ceteris paribus preferences

These preferences are similar to strong preferences. They also induce a unique partial order on outcomes. We can also use both construction principles used in optimistic and pessimistic semantics to compute a unique complete preorder.

```
Algorithm 3: A complete preorder associated with $\mathcal{P}_{>_{st}}$ .
Data: A preference set $\mathcal{P}_{>_{st}}$ .
Result: A complete preorder $\succeq$ on $\Omega$.
begin
    $l \leftarrow 0$
    while $\Omega \neq \emptyset$ do
        $l = l + 1$
        $E_l = \{t | t \in \Omega, \nexists (L(s_i), R(s_i)) \in \mathcal{L}(\mathcal{P}_{>_{st}}), t \in R(s_i)\}$
        if $E_l = \emptyset$ then
            stop (inconsistent preferences), $l = l - 1$
        - $\Omega = \Omega \backslash E_l$
        - replace $(L(s_i), R(s_i))$ by $(L(s_i) \backslash E_l, R(s_i))$
        /** remove satisfied preferences **/
        - remove $(L(s_i), R(s_i))$ where $L(s_i) = \emptyset$
    return $\succeq = (E_1, \cdots, E_l)$.
end
```

Example 6 *(Example 1 con'd) Following the gravitation towards the ideal we have $\succeq = (\{t_1\}, \{t_3, t_5\}, \{t_0, t_7\}, \{t_2, t_4\}, \{t_6\})$ while following the gravitation towards the worst we have $\succeq = (\{t_1\}, \{t_3\}, \{t_0\}, \{t_2, t_7\}, \{t_4, t_5, t_6\})$.*

Experimental Study

Our main objective is to evaluate the psychological plausibility of strong, optimistic, pessimistic and ceteris paribus semantics. In order to reach this objective, we have conducted a psychological experiment devoted to collect sets of comparative preferences formulated by participants to this experiment, and the associated models (a (pre)order on the set of outcomes). The adopted methodology and main results are presented in the next subsections.

Method

Participants Thirty first-year psychology students at the University of Toulouse-Le Mirail, all native French speakers, contributed to this study. None of them had previously received any formal logical training or any course on preferences. Note that our objective is not to study participant's real preferred menus. Such an objective would necessitate a much more large number of participants. Rather, our objective is to compare statistically the fit of the semantics under study with human preference's judgments. For such an objective, our sample size is sufficient according to scientific standards.

Material and procedure Comparative preference judgments were collected via a booklet where subjects were asked to suppose that they are at the restaurant and they must compose their menu. In the first page of the booklet, they were asked to compare and to rank-order the following objects (unranked objects where skipped from analyses):
t_0: fish-white-ice_cream, t_1: fish-white-cake
t_2: fish-red-ice_cream, t_3: fish-red-cake

[3] Technically speaking, this preorder can be obtained by min-based aggregation operator of all preorders satisfying the set of preferences.

$(white, red), (meat, fish), (ice\_cream, cake),$
$(meat - white - ice\_cream, meat - white - cake),$
$(meat - white - ice\_cream, meat - red - ice\_cream),$
$(meat - white - ice\_cream, meat - red - cake),$
$(meat - white - ice\_cream, fish - white - ice\_cream),$
$(meat - white - ice\_cream, fish - white - cake),$
$(meat - white - ice\_cream, fish - red - ice\_cream),$
$(meat - white - ice\_cream, fish - red - cake),$
$(meat - white - cake, meat - red - ice\_cream),$
$(meat - white - cake, meat - red - cake),$
$(meat - white - cake, fish - white - ice\_cream),$
$(meat - white - cake, fish - white - cake),$
$(meat - white - cake, fish - red - ice\_cream),$
$(meat - white - cake, fish - red - cake),$
$(meat - red - ice\_cream, meat - red - cake),$
$(meat - red - ice\_cream, fish - white - ice\_cream),$
$(meat - red - ice\_cream, fish - white - cake),$
$(meat - red - ice\_cream, fish - red - ice\_cream),$
$(meat - red - ice\_cream, fish - red - cake),$
$(meat - red - cake, fish - white - ice\_cream),$
$(meat - red - cake, fish - white - cake),$
$(meat - red - cake, fish - red - ice\_cream),$
$(meat - red - cake, fish - red - cake),$
$(fish - white - ice\_cream, fish - white - cake),$
$(fish - white - ice\_cream, fish - red - ice\_cream),$
$(fish - white - ice\_cream, fish - red - cake),$
$(fish - white - cake, fish - red - ice\_cream),$
$(fish - white - cake, fish - red - cake),$
$(fish - red - ice\_cream, fish - red - cake).$

Table 1: Pairs of menus participants have to compare.

t_4: meat-white-ice_cream, t_5: meat-white-cake
t_6: meat-red-ice_cream, t_7: meat-red-cake.
Next, they were asked to compare the 31 pairs of menus given in Table 1. An object o_1 can be preferred to an object o_2 or o_2 preferred to o_1, or be both equally preferred, or be incomparable. Answers of the kinds ”equally preferred” or ”incomparable” have been discarded from analysis.

Rationale Given participant's comparative preference judgments, for each participant, we computed the set of compact preferences (see Table 2) consistent with participant's preferences. For a given participant, a comparative preference is retained as compact if it is consistent with all her/his preferred menus (see Table 1).

Next, given these compact preferences and the algorithms provided in the paper, for each participant, four preorders have been inferred according to the principles underling the inferential machinery of the four studied semantics. For evaluating the psychological relevance of these semantics, the key comparison is between participant's (pre)order on the 8 menus $\{t_0, \cdots, t_7\}$ and (pre)orders computed according to the four semantics given participant's compact preferences. Two cues have been used for ordering semantics according to their psychological relevance: *The percentages of cases* where the semantics provide an inconsistent set of models; and *the distance* and *mean distance* between ranks allowed by participants and semantics to the 8 menus.

$white$	$> (vs. <)$	red
$meat$	$> (vs. <)$	$fish$
ice_cream	$> (vs. <)$	$cake$
$white \wedge meat$	$> (vs. <)$	$white \wedge fish$
$white \wedge ice\_cream$	$> (vs. <)$	$white \wedge cake$
$red \wedge meat$	$> (vs. <)$	$red \wedge fish$
$red \wedge ice\_cream$	$> (vs. <)$	$red \wedge cake$
$meat \wedge ice\_cream$	$> (vs. <)$	$meat \wedge cake$
$fish \wedge ice\_cream$	$> (vs. <)$	$fish \wedge cake$

Table 2: Set of a priori possible compact preferences.

- *Percentages of inconsistency:* For each semantics, we computed the percentages of cases where it produces an inconsistent set of models given inferred participant's compact preferences. A semantics better fits psychological data if it allows producing a consistent set of models from participant's compact preferences.
- *Mean Distances:* Two distances based on participants and semantics orders have been computed. In both cases, distances are computed from the ranks attributed to each of the 8 menus by participants and semantics. Several menus can have the same rank. Suppose participant 1 prefers the menu “meat, red wine, ice cream”, if this menu is also the preferred one for a given semantics, then the distance is zero. If only one menu is more preferred, then the rank is 1, and so on. A semantics better fits psychological data if the rank it attributes is closer to the menu preferred by participants. A semantics better fits psychological data if the mean distance between participants's preferred models and the rank attributed by the semantics is smaller. The same calculus can be made for each menu involved in participant's ranking (which doesn't necessarily involve the 8 proposed menus). So, for each participant, it is possible to compute the mean of the distance between each menu and ranks predicted by semantics. Next, the mean of these means is computed. As before, a small mean means a better fit.

In order to conclude at the inferential level, cognitive psychology, exactly as other experimental sciences, makes use of statistical tools for hypothesis testing. The student's t-test allows testing the null hypothesis that two means are not different. The probability p provided by the test express the risk (called alpha) that we reject by error the null hypothesis. In social and human sciences, it is usual to consider that this risk is acceptable at the level .05, that is, if p is greater than .05, we cannot reject the null hypothesis without a significant risk. Under .05, we reject the null hypothesis, and so accept the hypothesis of the difference between the two means. In our analyses, when a difference between means is significant ($p =< .05$), it is interpreted as: the seman-

	cp	$str.$	$pess.$	$opt.$
$\% \ inconsistency$	36.6	10	0	0
$Mean\ distances$ $to\ participants$ $peferred\ outcomes$ $(standard\ deviation)$	1.5 (.81) $n =$ 26	.83 (.93) $n =$ 29	.62 (.71) $n =$ 30	.55 (.65) $n =$ 30
$Means\ of\ the\ mean$ $distance\ to\ participants$ $outcome\ levels$ $(standard\ deviation)$ $n = 19$	2.44 (.63)	2.46 (.65)	2.54 (.71)	2.53 (.66)

Table 3: Cues for evaluation of the fit of semantics with participant's preference judgment. "cp", "str", "pess." and "opt." stand respectively for ceteris paribus, strong, pessimistic and optimistic.

tics exhibiting the less mean distance significantly fits better human data than the other semantics.

Results

Participant's answers allowed to compute a set of compact preferences containing between 3 and 7 compact preferences out of 18 a priori ones. Table 3 shows that the ceteris paribus semantics doesn't fit participant's orders in 36% of the cases and that the strong semantics failed in 10% of the cases, while optimistic and pessimistic semantics provide always a consistent set of preferences. This order is confirmed by comparisons of distances between participants and semantics' levels for participant preferred outcome. Table 3 also suggests that the optimistic semantics has a better fit than the pessimistic one (however mean's comparison by Student's t-test is not significant: $t = -1.43$, $df = 29$, $p = .16$) while the latter has a better fit than the strong semantics ($t = 2.7$, $df = 28$, $p = .01$) which better fits participant's data than ceteris paribus semantics ($t = -5.7$, $df = 24$, $p < .001$, significant). These results are broadly confirmed by the comparison of the means of the mean distance between participants and semantics (pre)orders. Indeed, statistical comparisons by Student's t-test show a significant difference between strong and pessimistic semantics ($t = -2.37$, $df = 18$, $p = .036$) but not between ceteris paribus and strong, and pessimistic and optimistic semantics. This result confirms that two distinct sets of semantics can be distinguished from their psychological relevance: Pessimistic and optimistic semantics on one hand, and strong and ceteris paribus on the other one. Except for percentages, more the values are low, better is the fit. As such, given all the information summarized in table 3, it appears that optimistic and pessimistic semantics are more plausible psychologically than strong and ceteris paribus semantics.

Conclusion

We focused on comparative preference statements and distinguished different non-monotonic semantics that have been studied in literature. So far, researchers have argued for a semantics or another from purely theoretical standpoint (also philosophical for ceteris paribus semantics) or for modeling a specific application. In this paper, we explored another dimension, namely psychological plausibility, to compare the semantics.

This work gives an indication about human behavior when interpreting comparative preferences. Our results suggest that pessimistic and optimistic semantics better fit human preferences organization and inference than ceteris paribus and strong semantics. Neverthless, it doesn't mean that every human in every situation would "prefer" according to the principles underling these semantics. Rather, it suggests that in familiar domains, a population known as representative of global occidental people, "prefer" in a manner more closed to pessimistic and optimistic semantics. Psychological plausibility is not of course the sole criterion for evaluating formal models in AI, but it is a criterion, every time a formal model could have incidences in human adaptation, including cognitive comfort and efficiency.

This first attempt opens the door to more ambitious and deeper comparison of preference representations. In a future work we intend to perform a comparison of the main different compact representations of preferences such as CP-nets (Boutilier et al. 2004), QCL (Brewka, Benferhat, and Le Berre 2004), etc.

References

Benferhat, S.; Dubois, D.; Kaci, S.; and Prade, H. 2002. Bipolar possibilistic representations. In *UAI'02*, 45–52.

Benferhat, S.; Bonnefon, J.; and Da Silva Neves, R. 2004. An experimental analysis of possibilistic default reasoning. In *KR'04*, 130–140.

Benferhat, S.; Dubois, D.; and Prade, H. 1992. Representing default rules in possibilistic logic. In *KR'92*, 673–684.

Boutilier, C.; Brafman, R.; Domshlak, C.; Hoos, H.; and Poole, D. 2004. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research* 21:135–191.

Boutilier, C. 1994. Toward a logic for qualitative decision theory. In *KR'94*, 75–86.

Brewka, G.; Benferhat, S.; and Le Berre, D. 2004. Qualitative choice logic. *Artificial Intelligence* 157(1-2):203–237.

Hansson, S. 1996. What is ceteris paribus preference? *Journal of Philosophical Logic* 25:307–332.

Kraus, S.; Lehmann, D.; and Magidor, M. 1990. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence* 44(1-2):167–207.

Neves, R. D. S.; Bonnefon, J.; and Raufaste, E. 2002. An empirical test of patterns for nonmonotonic inference. *Annals of Mathematics and Artificial Intelligence* 34(1-3):107–130.

Pearl, J. 1990. System Z: A natural ordering of defaults with tractable applications to default reasoning. In *TARK'90*, 121–135.

von Wright, G. H. 1963. *The Logic of Preference*. University of Edinburgh Press.

Yager, R. 1983. Entropy and specificity in a mathematical theory of evidence. *International Journal of General Systems* 9:249–260.

Modelling Cryptographic Protocols in a Theory of Action

James P. Delgrande and **Torsten Grote** and **Aaron Hunter**
School of Computing Science,
Simon Fraser University,
Burnaby, B.C.,
Canada V5A 1S6.
{jim,tga14,hunter}@cs.sfu.ca

Abstract

This paper proposes a framework for analysing cryptographic protocols by expressing message passing and possible attacks as a situation calculus theory. While cryptographic protocols are usually quite short, they are nonetheless notoriously difficult to analyse, and are subject to subtle and nonintuitive attacks. Our thesis is that in previous approaches for expressing protocols, underlying domain assumptions and capabilities of agents are left implicit. We propose a declarative specification of such assumptions and capabilities in the situation calculus. A protocol is then compiled into a sequence of actions to be executed by the principals. A successful attack is an executable plan by an intruder that compromises the stated goal of the plan. We argue that not only is a full declarative specification necessary, it is also much more flexible than previous approaches, permitting among other things interleaved runs of different protocols and participants with varying abilities.

Introduction

A cryptographic protocol is a formalised sequence of messages between agents, where parts of a message are protected using cryptographic functions such as encryption. These protocols are used for many purposes, including the secure exchange of information, carrying out a transaction, authenticating an agent, etc. Protocols are typically specified in the following format:

The Challenge-Response Protocol

1. $A \rightarrow B : \{N_A\}_{K_{AB}}$
2. $B \rightarrow A : N_A$

In this protocol, the goal is for agent A to determine whether B is alive on the network. The first step is for A to send B the message N_A encrypted with a shared key K_{AB}. N_A is a *nonce*, a random number assumed to be new to the network. The second step is for B to send A the message N_A unencrypted. Since only A and B have K_{AB}, and K_{AB} is assumed to be secure, it would seem that N_A could only have been decrypted by B, and so B must be alive. However, the protocol is flawed; here is an attack:

An Attack on the Challenge-Response Protocol

1. $A \rightarrow I_B : \{N_A\}_{K_{AB}}$
1.1 $I_B \rightarrow A : \{N_A\}_{K_{AB}}$
1.2 $A \rightarrow I_B : N_A$
2. $I_B \rightarrow A : N_A$

An intruder I intercepts the message in line 1 and, masquerading as B, initiates a round of the protocol with A, thereby obtaining the decrypted nonce.

While this example is simplistic, it illustrates the type of problems that arise in protocol verification. Even though protocols are usually short, they are notoriously difficult to prove correct. As a result, many different formal approaches have been developed for protocol verification. In these approaches, a protocol is generally specified as above, and then one tries to develop an attack on the protocol. However, often these approaches are difficult to apply by anyone other than the original developers (Brackin, Meadows, & Millen 1999). Part of the problem is that there is no clear agreement on exactly what an attack really is (Aiello & Massacci 2001), which leaves considerable ambiguity about the status of a protocol when no attack is found. Moreover, as we later discuss, the language for specifying a protocol is highly ambiguous, and much information is left implicit. Thus it is no surprise that protocols are hard to convincingly prove secure.

Our thesis is that all aspects of a protocol need to be explicitly specified, and moreover that protocol verification may profitably be viewed as a problem in commonsense reasoning and agent communication. The main contribution of this paper is the introduction of a declarative, commonsense theory of message passing between agents, suitable for proving results about protocols, expressed as a situation calculus theory. The framework makes explicit background assumptions, protocol goals, agent's capabilities, and the message passing environment. A protocol is *translated* into a set sequence of actions for agents to execute. These actions may be interleaved with others, and the framework allows simultaneous runnings of multiple protocols. The aim of an intruder is to construct a *plan* such that the goal of the protocol, in a precise sense, is thwarted. A protocol is secure when no such plan is possible. A valid protocol then is one which is secure, which may complete, and in which at completion the goal is provably established. The approach is flexible, and significantly more general than previous approaches since we can tailor the agents and the environment to specific applications. For example, we can model intruders with different capabilities and we can model several different protocols running at the same time. This work is intended as the first step towards a new automated verification system for protocols based on a language such as ConGolog.

The next section briefly introduces work in cryptographic protocol verification. The third section motivates our approach to the problem, while the following section presents an axiomatisation of an instance of the approach in the situation calculus. The last section sketches contributions and future work.

Related Work

The standard intruder model is of a very powerful adversary, the so-called *Dolev-Yao intruder* (Dolev & Yao 1983). Informally, the intruder can read, block, intercept, or forward any message sent by an honest agent. Hence, a message recipient is never aware of the identity of the sender, except possibly via encrypted messages. The first logic-based approach to protocol verification was the *BAN logic* of (Burrows, Abadi, & Needham 1990). The logic is rather ad hoc, as it consists of a set of rules of inference with no formal semantics. However, it has been highly influential because it illustrates the importance of *knowledge* in protocol verification and also because it illustrates how protocol verification can be reduced to reasoning in a formal logical system.

One standard formal tool for reasoning about the knowledge of several agents is the multi-agent systems framework of (Fagin *et al.* 1995). In protocol verification, the *strand space* formalism provides a similar model of message passing between several agents (Thayer, Herzog, & Guttman 1999). A strand space is a formal representation of all possible traces corresponding to runs of a specified protocol; it enables a protocol analyzer to show that an intruder cannot compromise a secure protocol. It has been proven that strand spaces are actually less expressive than multi-agent systems (Halpern & Pucella 2003). One notable weakness is that the framework does not provide a suitable model of knowledge.

Formal tools developed for knowledge representation and reasoning have also been used for protocol verification. One such tool is logic programming under the stable model semantics (Gelfond & Lifschitz 1991). Cryptographic protocols have been encoded as logic programs where the stable models correspond to attacks that an intruder can perform (Aiello & Massacci 2001). There are at least two issues with the encodings. First, the logic program must be handcrafted for each protocol to be analyzed. Second, the attack must be specified in advance; new attacks are not detected automatically. (Wang & Zhang 2008) proposes a very similar approach. Neither approach is elaboration tolerant, and neither intensional.

In related work, protocols have been represented in a multi-set rewriting formalism, and then translated into the same logic programming paradigm used in the previous two approaches (Armando, Compagna, & Lierler 2004). Instead of a model checker, this translation is solved with an answer set solver, acting as an alternative back-end to the protocol verification tool AVISPA[1]. To date, this translation approach has not proven to be practical.

Hernández and Pinto propose an approach that is similar to ours; in particular they also use the situation calculus (Hernández-Orallo & Pinto 1997). However, they focus on producing proofs of correctness based on the actions of honest agents. In contrast, we explicitly model the actions of an intruder, and we view protocol verification as communicating while guarding against attack. Our treatment of the communication channel is also different: while Hernández and Pinto define an unreliable broadcast channel, we define a direct channel that allows the intruder the first opportunity to receive a message. As such, our approach is best understood as addressing a somewhat different problem than the Hernández-Pinto approach.

There has of course been extensive work in reasoning about action. Due to space limitations, we assume a familiarity with the situation calculus (Levesque, Pirri, & Reiter 1998), and we assume Reiter's solution to the frame problemn without further comment. We note that other action formalisms would have worked equally well in formalising the approach.

Motivation

Consider the Challenge-Response protocol and the attack described previously. Several things may be noted about the protocol specification. First, while the intent of the protocol and the attack are intuitively clear, the meaning of the exchanges in the protocol are ambiguous. Consider the first line of the protocol: it cannot mean that A sends a message to B, since this may not be the case, as the attack illustrates. Nor can it mean that A *intends* to send a message to B, because in the attack it certainly isn't A's intention to send the message to the intruder! Moreover, there is more than one action taking place in the first line, since A sends a message and B is involved in the (potential) receipt of a message. Hence, the specification language is inexpressive; notions of agent communication should be made explicit.

As well, the specification leaves important aspects of the problem unstated. For instance, it is not stated that the goal of the protocol is to convince A that B is alive. Nor is it stated how this goal is to be accomplished, in this case indirectly via the encryption and sending of messages. Metalevel reasoning is required to determine if a protocol is secure, or if an attack on the protocol is possible. So notions of protocol goal and attack should also be made explicit.

The protocol specification also does not state the fact that N_A is a freshly generated nonce, nor the fact that the key K_{AB} is only known to A and B. Moreover, the capabilities of agents are not specified. For example, the intruder is assumed to be able to intercept and redirect messages; however it can decrypt a message only if has the appropriate key.

Last, there is no recognition that a protocol execution will take place in a broader context that includes other agent actions and other protocol executions. Nor does it take into account the interleaving of actions with the execution of a given instance of a protocol. For example, it is quite possible that a protocol could fail via what might be called a "stupidity attack". Consider the following exchange:

Another Attack on the Challenge-Response Protocol

1. $A \rightarrow I_B : \{N_A\}_{K_{AB}}$
1.1 $A \rightarrow I_B : N_A$
2. $I_B \rightarrow A : N_A$

[1] http://avispa-project.org/

In this case A sends the unencrypted nonce to the intruder. This of course is outlandish, but it nonetheless represents a logically possible compromise of the protocol (and in fact any other "secure" protocol). The point is that, much like the qualification problem in planning, there is an assumption that "nothing untoward happens" in a protocol execution. However, it may well be that there are "untoward happenings" much more subtle than the stupidity attack; consequently, it is desirable to have a framework for specifying protocols that is general enough to take such possibilities into account.

We argue that in order to provide a robust demonstration of the security and correctness of a protocol, all of the above points need to be addressed. We suggest that an explicit, logical formalisation in the situation calculus provides a suitable framework. Broadly speaking, our primary aim is to clearly formalize exactly what is going on in a cryptographic protocol in a declarative action formalism; such a formalization will provide a more flexible model of agent communication.

Approach

We present an outline of a formalization for cryptographic protocols, using the Challenge-Response protocol as an example. While we don't completely cover all points raised in the previous section, given space constraints, it should be clear that any omissions are easily addressable.

Vocabulary

We formalize message passing systems in the situation calculus. For our purposes, there are four main sorts of objects (beyond actions and situations): *agents, keys, messages* and *nonces*. In this section, we briefly describe each sort.

Agents: The term *agent* refers to both honest agents and to the malicious intruder. We reserve the term *principal* to refer to an honest agent. Variables a, a_1, ... range over agents. The constant $intr$ denotes the intruder. Unary predicates $Agent$ and $Intruder$ have their obvious meanings.

Fluent $Alive(a, s)$ indicates that a is alive in situation s. It is a precondition for executing any action; for brevity however we omit it in action preconditions. $Has(a, x, s)$ means that a has access to x in situation s, where the variable x ranges over messages, keys and nonces. This can be seen as a kind of knowledge, but we use the epistemically neutral term Has and interpret the meaning in terms of "access" to information. We use $Bel(a, f, s)$ to indicate that a believes that the fluent f is true in situation s. The semantics of Bel can be defined using the treatment of belief in (Scherl & Levesque 2003) (where they use $Knows$ for Bel).

Messages: Communication in our framework involves the exchange of messages. Variables $m, m_1, \ldots$ range over messages. Unary predicate Msg is true of messages. Messages are considered to be atemporal, and so are not indexed by a situation. Messages are composed of a finite sequence of parts, which may be nonces, agent names, or keys; each part may be encrypted. We assume an appropriate situation calculus axiomatization of lists, including the constructor $list(p_1, \ldots, p_n)$ and selectors $first(m)$, $second(m)$, etc. A useful state constraint[2] is that if an agent Has a message, then it has the message parts, for example:

$$Has(a, m, s) \wedge Msg(m) \supset Has(a, first(m), s).$$

Keys: Variables k, k_1, ... range over keys. Predicate $Key(k)$ indicates that k is a key, while $SymKey(k)$ and $AsymKey(k_1, k_2)$ have their expected meaning for symmetric and asymmetric keys respectively. $ShKey(a_1, a_2, k)$ indicates that k is a shared (symmetric) key for agents a_1, a_2. $PubKey(a, k)$ and $PrivKey(a, k)$ give public and private keys, respectively, of an agent.

Three functions are associated with keys: The value of $encKey(x)$ is the key which has been used to encrypt x. The value of $enc(x, k)$ is the result of encrypting x with k; and $dec(x, k)$ returns the corresponding decrypted message. The following state constraint relates enc and $encKey$; others (omitted here) relate public and private keys, etc.:

$$m_1 = enc(m_2, k) \supset k = encKey(m_1).$$

Nonces: Variables $n, n_1, \ldots$ range over nonces. The most important feature of nonces is that they must be *freshly generated* during the current protocol run. The fluent $IsFresh(n, s)$ is intended to be true if and only if the nonce n has been generated "recently" with respect to the situation s. To this end, the functional fluent $fresh(s)$ is used to model the generation of new nonces during a protocol run using the axiom $fresh(s) = fresh(s') \supset s = s'$.

Actions: There are two classes of action terms. The class of *basic actions* is comprised of actions for encryption and decryption, sending and receiving messages, and composing messages. These actions are described next. *Protocol-specific actions* are described later, in the section on representing a protocol in an action theory.

To ease readability we omit sort predicates. The variable conventions given above implicitly specify the sort of each variable. As usual, free variables are implicitly universally quantified.

1. $encrypt(a, x, k)$ – Agent a encrypts nonce or message x using key k.

   Precondition:

   $Poss(encrypt(a, x, k), s) \equiv (Has(a, x, s) \wedge (Has(a, k, s) \vee \exists a' PublicKey(a', k)))$

   Effect:

   $Has(a, enc(x, k), do(encrypt(a, x, k), s))$

2. $decrypt(a, x, k)$ – Agent a decrypts x using key k.

   Precondition:

   $Poss(decrypt(a, x, k), s) \equiv (Has(a, x, s) \wedge Has(a, k, s) \wedge [(SymKey(k) \wedge k = encKey(x)) \vee (AsymKey(k, k') \wedge k' = encKey(x))])$

[2] State constraints can be problematic, and are not part of a *basic action theory* (Reiter 2001). Nonetheless they are useful in a representational context, in initially specifying a theory.

Effect:

$Has(a, dec(x,k), do(decrypt(a,x,k), s))$

3. $send(a_1, a_2, m)$ – Agent a_1 sends m intended for a_2. The intruder can masquerade as the sender. Fluent $Sent$ indicates that a message is in some fashion “posted”, that is can be received by an agent.

   Precondition:

   $Poss(send(a_1, a_2, m), s) \equiv$
   $((Has(a_1, m, s) \wedge a_1 \neq a_2) \vee Has(intr, m, s))$

   Effect:

   $Sent(a_1, a_2, m, do(send(a_1, a_2, m), s))$

4. $receive(a_1, a_2, m)$ – a_1 receives message m from a_2. The intruder can intercept messages. $\neg Sent$ indicates that the message is no longer available to be received.

   Precondition:

   $Poss(receive(a_1, a_2, m), s) \equiv (Sent(a_2, a_1, m, s) \vee$
   $(a_1 = intr \wedge \exists a' \, Sent(a_2, a', m, s)))$

   Effect:

   $Has(a_1, m, do(receive(a_1, a_2, m), s)) \wedge$
   $\neg Sent(a_2, a_1, m, do(receive(a_1, a_2, m), s)) \wedge$
   $Recd(a_1, a_2, m, do(receive(a_1, a_2, m), s))$

5. $compose(a, m, x)$ – Agent a composes message m having body x.

   Precondition:

   $Poss(compose(a, m, list(x_1, \ldots x_n)), s) \equiv$
   $(Has(a, x_1, s) \wedge \ldots Has(a, x_n, s))$

   Effect:

   $Has(a, m, do(compose(a, m, list(x_1, \ldots x_n)), s)) \wedge$
   $Msg(m) \wedge first(m) = x_1 \wedge second(m) = x_2 \wedge \ldots$

   Since messages in a protocol always have fixed length, an alternative is to have *compose* take message parts as arguments. Thus there would be a set of *compose* actions, one for each possible message length.

State Constraints

Some state constraints have already been mentioned. For proving properties about protocols, some *epistemic* constraints are useful, for example, an agent knows what actions it carried out. In the Challenge-Response protocol we use the following:

$$Sent(a_1, a_2, m, s) \supset Bel(a_1, Sent(a_1, a_2, m), s)$$
$$Recd(a_1, a_2, m, s) \supset Bel(a_1, Recd(a_1, a_2, m), s)$$

We can then state that if an agent a_1 sends a fresh nonce encrypted in the key it shares with a_2, and gets the unencrypted nonce back, then a_1 believes that a_2 is alive:

$$(Bel(a_1, Sent(a_1, a_2, en), s) \wedge en = enc(n,k) \wedge Fresh(n) \wedge ShKey(a_1, a_2, k) \wedge Bel(a_1, Recd(a_1, x, n), s)) \supset Bel(a_1, Alive(a_2), s)$$

Initial Situation

The initial situation contains information about the number of agents, their keys, etc. Since the details are straightforward, we just outline what is required. For example, using the $Agent$ predicate, a finite set of principals is specified, along with the intruder, $intr$. For each agent, we specify a combination of private, public, and shared keys. Typically, this is all we need to specify in the initial situation.

Adding Control Constraints

Parallelism is simulated by allowing concurrent interleaving of actions. We model a Dolev-Yao intruder through the following scheme, which allows the intruder to perform an arbitrary number of actions before an honest agent can act:

```
loop {
  Intruder executes some actions;
  A principal executes one action
}
```

This can be implemented in our action theory as follows; assume that fluent OkP isn't used in the theory. Informally OkP states that it is ok for a principal to execute an action. Basic actions are modified as follows:

- For a principal: Each precondition $Poss(a, s) \equiv \phi$ is modified to $Poss(a, s) \equiv (\phi \wedge OkP(s))$. Each effect axiom $\psi(do(a, s))$ is replaced by $\psi(do(a, s)) \wedge \neg OkP(do(a, s))$.
- Only the intruder can make $OkP(s)$ true. A new action $onOkP$ is introduced with precondition $Poss(onOkP(a), s) \equiv (a = intr)$ and effect $OkP(do(onOkP, s))$.

An advantages of this framework is that other models of concurrency can be easily expressed. For example, it is straightforward to specify that the intruder may carry out one action, followed by some agent carrying out an action. In this case, the intruder is limited in that it may not be able to compromise all protocol runs. On the other hand, there are some principal actions that an intruder cannot compromise, such as encryptions and decryptions. So from an efficiency standpoint it would make sense to allow an agent to execute a full sequence of such “uncompromisable” actions. To this end, a full implementation could make use of higher-level imperative constructs, such as a sequence of actions as given in Golog's *Do* (Levesque *et al.* 1997).

Representing a Protocol in an Action Theory

The goal of the preceding framework is to completely and explicitly specify a theory of agent communication involving encryption, freshly generated nonces, and a hostile intruder. In this setting, a protocol is regarded as a high-level description of prescribed agent actions, designed to achieve some goal in a dynamic, unpredictable, hostile environment. Hence there are two things that remain to be specified:

1. how the protocol corresponds to sets of agent actions, and
2. the goal of the protocol.

Compiling a Protocol into an Action Theory Our goal is to express a protocol such as the Challenge-Response protocol in terms of our action theory. Our ultimate goal is to *automate* this process, so that any protocol can be translated and integrated with our situation calculus theory. Hence the ultimate goal is to provide a *compiler* for protocols into action theories. At present we hand code a translation, giving the Challenge-Response protocol as an example below. We suggest via this example that a specification of a translator presents no great technical difficulty.

There are two general methodologies for translating a protocol specification into our action theory, corresponding to two levels of granularity:

1. Compile lines of a protocol into new, protocol-specific actions.
2. Compile each line of a protocol into two sequences of previously-defined, basic actions. The first sequence captures the implicit composition and sending of a message; while the second captures the implicit receipt and decrypting of a message.

We are currently implementing the first approach. Each line of a protocol is implicitly made up of two parts, the first involving the composition and sending of a message, and the second involving the receiving and decrypting of the message. Thus in the first line of the Challenge-Response protocol, the intent is that A compose a message and send it, followed by B receiving it and decrypting it. However, note that for every pair of successive lines in a protocol, the implicit $receive$ of one line can be combined with the $send$ of the next. Thus in the Challenge-Response protocol, B's receiving of a message from A can be combined with a sending of an unencrypted nonce back to A. Hence a n-line protocol can compile into $n+1$ protocol-specific actions – one for the first line of the protocol, one for the last line, and one for each of the $n-1$ successive pair of lines. Thus the Challenge-Response protocol would compile into three new protocol-specific actions:

$CR.1.send$**:** Agent a_1 composes a message with a fresh nonce, encrypted in the key shared with a_2, and sends it to a_2.

$CR.1.rec.2.send$**:**[3] a_2 receives the message, decrypts it, and sends a message with the nonce to a_1.

$CR.2.rec$**:** a_1 receives the unencrypted nonce from a_2.

We introduce the following constants and fluents: $\langle pid\rangle$ is an identifier inserted by the compiler giving the protocol type and instance of the run. (We also use pid without angle brackets as a variable.) Predicate $Type$ extracts the protocol type from its argument; here $Type(pid) = \text{“}CR\text{”}$. Fluent $Expect$ expresses control knowledge, that after initiating a run of the protocol, a_1 expects at some point to receive a message from a_2 comprising the second step in this instance of the protocol. In this way, multiple instances of multiple protocols may be concurrently executed. Fluent $Completed$ indicates that the protocol has completed successfully.

We have the following action preconditions and effects:

$CR.1.send$:

Precondition:

$$\begin{aligned}&Poss(CR.1.send(a_1,a_2,m,k,n),s) \equiv\\ &\quad ShKey(a_1,a_2,k) \wedge n = fresh(s) \wedge\\ &\quad m = list(\langle pid\rangle, enc(n,k))\end{aligned}$$

Effect: Let $s' = do(CR.1.send(a_1,a_2,m,k,n),s)$.

$$\begin{aligned}&Sent(a_1,a_2,m,s') \wedge Has(a_1,m,s') \wedge Has(a_1,n,s') \wedge\\ &\quad Has(a_1,enc(n,k),s') \wedge Expect(a_1,a_2,\langle pid\rangle,2,s')\end{aligned}$$

$CR.1.rec.2.send$:

Precondition:

$$\begin{aligned}&Poss(CR.1.rec.2.send(a_2,a_1,m,m'),s) \equiv\\ &\quad Sent(a_1,a_2,m,s) \wedge Type(first(m)) = \text{“}CR\text{”} \wedge\\ &\quad Has(a_2,encKey(m),s) \wedge\\ &\quad m' = list(first(m), dec(second(m),encKey(m)))\end{aligned}$$

The precondition is cumbersome, reflecting the fact that several actions (including a receive and send) are combined into one protocol-specific action.

Effect: Let $s' = do(CR.1.rec.2.send(a_2,a_1,m,m'),s)$.

$$\begin{aligned}&Recd(a_2,a_1,m,s') \wedge Has(a_2,m,s') \wedge\\ &\quad Has(a_2,first(m),s') \wedge Has(a_2,second(m),s') \wedge\\ &\quad Has(a_2,dec(second(m),encKey(m)),s') \wedge\\ &\quad \neg Sent(a_1,a_2,m,s') \wedge Sent(a_2,a_1,m',s')\end{aligned}$$

The effect is likewise cumbersome: a_2 has the message and all its parts; the original message is marked as unavailable; and a new message is sent to a_1.

$CR.2.rec$:

Precondition:

$$\begin{aligned}&Poss(CR.2.rec(a_1,a_2,m),s) \equiv\\ &\quad Sent(a_2,a_1,m,s) \wedge Type(first(m)) = \text{“}CR\text{”} \wedge\\ &\quad Expect(a_1,a_2,first(m),2,s)\end{aligned}$$

Effect: Let $s' = do(CR.2.rec(a_1,a_2,m),s)$.

$$\begin{aligned}&Recd(a_1,a_2,m,s') \wedge Has(a_1,m,s') \wedge\\ &\quad Has(a_1,first(m),s') \wedge Has(a_1,second(m),s') \wedge\\ &\neg Sent(a_2,a_1,m,s') \wedge Completed(a_1,a_2,first(m),s')\end{aligned}$$

Expressing the Goal of a Protocol The goal of a protocol will often have epistemic components. For the Challenge-Response protocol, the overall goal is that if a protocol run successfully completes, then the initiating agent will believe that the responding agent is alive; and moreover, it is not possible that the initiating agent believe that the second agent is alive when in fact it is not. (That is, the initiating agent's belief is indeed knowledge.)

$$\begin{aligned}&(Completed(a_1,a_2,x,s) \wedge Type(x) = \text{“}CR\text{”}) \supset\\ &\quad (Bel(a_1,Alive(a_2),s) \equiv Alive(a_2,s))\end{aligned}$$

This assumes that principals are alive or dead on the network, independent of the situation. A more nuanced representation would take into account the possibility that an agent may become not $Alive$.

There are other parts to a successful protocol specification that need to be specified. First, it must be possible for there to be a successful run:

$$\exists s.\ Completed(a_1,a_2,x,s) \wedge Type(x) = \text{“}CR\text{”}$$

That is, a protocol that can never complete will vacuously

[3] The naming here is awkward, but is intended to be mnemonic for the protocol name (CR), along with the receive part of one line ($1.rec$) and the send part of the next ($2.send$).

never be compromised, but is of no use. Second, it would be desirable to prove that if the intruder carries out no actions, then the protocol is guaranteed to succeed.

The Attack on the CR Protocol

We now illustrate the approach by describing the attack on the Challenge-Response protocol:[4]

1. Agent a_1 initiates a round of the protocol with action:
   $CR.1.send(a_1, a_2, (\text{“}CR\text{”}, enc(n, k)), k, n)$
   One effect is $Sent(a_1, a_2, (\text{“}CR\text{”}, enc(n, k)))$
2. The intruder intercepts the sent message:
   $receive(intr, a_2, (\text{“}CR\text{”}, enc(n, k)))$
3. The intruder sends a message to a_1, masquerading as a_2:
   $send(a_2, a_1, (\text{“}CR\text{”}, enc(n, k)))$
4. The message is received by a_1 who understands it as an initiation of a new round of the CR protocol by a_2, and so responds with:
   $CR.1.rec.2.send(a_1, a_2, (\text{“}CR\text{”}, enc(n, k)), (\text{“}CR\text{”}, n))$
   This action has an effect $Sent(a_1, a_2, (\text{“}CR\text{”}, n))$.
5. The intruder intercepts this message:
   $receive(intr, a_1, (\text{“}CR\text{”}, n))$.
   This has effects $Has(intr, (\text{“}CR\text{”}, n))$, $Has(intr, n)$.
6. The intruder sends the nonce to a_1, masquerading as a_2:
   $send(a_2, a_1, (\text{“}CR\text{”}, n))$
7. The message is received by a_1:
   $CR.2.rec(a_1, a_2, (\text{“}CR\text{”}, n))$
   a_1 understands it as the completion of the original protocol; thus a_1 believes a_2 alive in the resulting situation.

Discussion

Thus far our focus has been on the development of an appropriate situation calculus formalization of cryptographic protocols. Our formalism is highly elaboration tolerant, in the sense that it is easy to axiomatize agents and intruders with different capabilities. For example, if we had information about the topology of a particular network, it would be easy to restrict an intruder to only intercept messages between particular principals. In most existing logical approaches to protocol verification, it is not straightforward to modify agent capabilities for a specific application.

In many cases, proofs of protocol correctness rely on the assumption that honest agents do not perform actions that compromise secret information; however, it is not always clear which actions are safe in this sense. In our framework, we can discover these undesirable actions and we can formally specify axioms that restrict honest agents from performing them. To the best of our knowledge, this problem has not been addressed in related formalisms.

There are two natural directions for future work on our framework. First, as noted previously, we would like to be able to directly compile protocol specifications into situation calculus theories. At present, we perform this encoding by hand; it would be desirable to automate this process, thereby facilitating the analysis of a wider range of protocols. The second direction is to implement a system for automatically finding attacks based on our situation calculus formalization. Our intention is to implement the system using ConGolog.

References

Aiello, L., and Massacci, F. 2001. Verifying security protocols as planning in logic programming. *ACM Transactions on Computational Logic* 2(4):542–580.

Armando, A.; Compagna, L.; and Lierler, Y. 2004. Automatic compilation of protocol insecurity problems into logic programming. In Alferes, J., and Leite, J., eds., *JELIA'04*, volume 3239 of *LNAI*, 617–627.

Brackin, S.; Meadows, C.; and Millen, J. 1999. CAPSL interface for the NRL protocol analyzer. In *Proceedings of ASSET 99*. IEEE Computer Society Press.

Burrows, M.; Abadi, M.; and Needham, R. 1990. A logic of authentication. *ACM TOCS* 8(1):18–36.

Dolev, D., and Yao, A. 1983. On the security of public key protocols. *IEEE Transactions on Information Theory* 2(29):198–208.

Fagin, R.; Halpern, J. Y.; Moses, Y.; and Vardi, M. Y. 1995. *Reasoning about Knowledge*. Cambridge, MA: The MIT Press.

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and deductive databases. *New Generation Computing* 9:365–385.

Halpern, J., and Pucella, R. 2003. On the relationship between strand spaces and multi-agent systems. *CoRR* cs.CR/0306107.

Hernández-Orallo, J., and Pinto, J. 1997. Formal modelling of cryptographic protocols in situation calculus. (Published in Spanish as: Especificación formal de protocolos criptográficos en Cálculo de Situaciones, *Novatica*, 143, pp. 57-63, 2000).

Levesque, H.; Reiter, R.; Lin, F.; and Scherl, R. 1997. Golog: A logic programming language for dynamic domains. *Journal of Logic Programming* 31.

Levesque, H.; Pirri, F.; and Reiter, R. 1998. Foundations for the situation calculus. *Linköping Electronic Articles in Computer and Information Science* 3(18).

Reiter, R. 2001. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. Cambridge, MA: The MIT Press.

Scherl, R., and Levesque, H. 2003. Knowledge, action, and the frame problem. *Artificial Intelligence* 144(1-2):1–39.

Thayer, F.; Herzog, J.; and Guttman, J. 1999. Strand spaces: Proving security protocols correct. *JCS* 7(1).

Wang, S., and Zhang, Y. 2008. A logic programming based framework for security protocol verification. In *Proc. ISMIS 2008*, volume 4994 of *LNAI*, 638–643. Springer.

[4] We suppress situation arguments in fluents for readability.

On Joint Ability in the Presence of Sensing

Hojjat Ghaderi and Hector Levesque
Department of Computer Science
University of Toronto
Toronto, ON M5S 3G4, Canada
{hojjat,hector}@cs.toronto.edu

Yves Lespérance
Department of Computer Science and Engineering
York University
Toronto, ON M3J 1P3, Canada
lesperan@cse.yorku.ca

Abstract

A central problem in the analysis of teams of agents is to assess when a group of autonomous agents, who may have private beliefs and goals, know enough together to be able to achieve a goal, should they so desire. In this paper, we present a definition of joint ability in the presence of sensing. We show through some simple examples involving private and public actions that it makes appropriate predictions with respect to coordination.

1 Introduction

An individual agent can often achieve a goal even if he does not initially know all the steps to follow, as long as he can *sense* along the way enough information to know what to do next, until the goal is attained. Clearly, one may delegate a goal only to an agent that is able to achieve it. Moore [1985] and others [Davis, 1994; Lespérance *et al.*, 2000] developed logical accounts of single agent *ability* in this sense.

Given this, an obvious question is how to extend this notion to *teams* of agents: when does a group know enough together, despite any incomplete knowledge or even false beliefs that they may have about the world or each other, to be *jointly able* to achieve a goal. Crucially, the agents need to know enough to stay coordinated. Unlike in the single-agent case, the mere existence of a joint plan mutually believed to achieve the goal (as in [Wooldridge and Jennings, 1999]) is not sufficient, since there may be several incompatible working plans and the agents may not be able to choose a share that coordinates with those of the others.

The issue of coordination has been thoroughly explored in game theory [Osborne and Rubinstein, 1999]. However, a major limitation of the classical game theoretic framework is that it assumes that there is a *complete specification* of the structure of the game including the beliefs of the agents. It is often also assumed that this structure is common knowledge among the agents. Recent work on the symbolic logic of games allows more incomplete and qualitative specifications, and supports symbolic reasoning over very large state spaces. However, most of this work, such as Coalition Logic [Pauly, 2002] and ATEL [van der Hoek and Wooldridge, 2003], is propositional, which limits expressiveness. More importantly, it ignores the issue of *coordination* within a coalition and simply assumes that a team can achieve a goal if there exists a strategy profile (or joint plan) over the agents that achieves it. This is only sufficient if we assume that the agents can communicate arbitrarily to coordinate as needed.

Ghaderi *et al.* [2007] proposed a logical framework to model the coordination of teams of agents based on the situation calculus. Their formalization avoids both of the limitations mentioned above: it supports reasoning on the basis of very incomplete specifications about the belief states of the agents, and it does not trivialize the issue of coordination. The formalization involves iterated elimination of dominated strategies [Osborne and Rubinstein, 1999]. Each agent eliminates strategies that are not as good as others given her private beliefs about the world and about what strategies the other agents would eliminate. This elimination process is repeated until it converges to a set of *preferred strategies* for each agent. Joint ability is said to hold if all combinations of preferred strategies succeed in achieving the goal.

However, Ghaderi *et al.* only considered example domains involving ordinary "world changing" actions. In this paper, we extend their account and show that it correctly handles domains with *sensing actions*, actions that allow an agent to obtain information by observing some aspect of the environment. We use an account of sensing actions and their effects on knowledge from [Shapiro *et al.*, 1998] and show that the techniques proposed by Ghaderi *et al.* to prove joint ability or the lack of it can be generalized to deal with sensing actions. This is significant, as it requires dealing with knowledge change, and strategies that may branch depending on what is learned by sensing. The space of strategies quickly becomes extremely large and it is significant that our symbolic proof techniques nonetheless allow results to be proven, even with only incomplete specifications of the agents' knowledge. Note that our approach can be also used to handle basic "informing" communication actions where the value of a fluent is communicated by one agent to another. These actions work very much like sensing, the difference being that it is the recipient of the communication that gets the new information.

In the next section, we describe a simple setup to test these ideas involving a safe with two locks. We then present the formalization of this domain and the definition of joint ability in the situation calculus. We then show the kind of ability results we can obtain in this formalization. Finally, we summarize our contributions and discuss future work.

2 Opening A Safe Together

Perhaps the simplest non-trivial example of ability in the single agent case was presented by Moore [1985]. He presents the example of a safe that can be opened by dialing the correct combination. Imagine that the safe explodes or jams or turns on a security alarm when any other number is dialed. Suppose there is an agent who does not know the combination of the safe. Although a plan surely exists to open the safe and the agent knows this, we would say that the agent is not able to open the safe. But if the correct combination is written on a piece of paper that the agent can pick up and read, we would now say that the agent is able to open the safe. This, in its most basic form, shows how ability depends on knowledge which itself depends on the available sensing actions.

To illustrate our formalization of joint ability, we will use a series of examples based on a two-person safe. The idea, roughly, is that two combinations are needed to open the safe, so that P and Q may be able to open the safe together even though neither one may know enough to do it alone. We also want to consider variants where, for example, Q knows both combinations allowing him to open the safe alone if P does not interfere, although P might not see this and ruin the plan.

To focus on the essential details only, we make a few simplifications. First of all, safe combinations will be binary: the safe has two locks A and B, and each lock has two buttons, 0 and 1. To open the safe, the correct button for lock A must first be pushed (using action *pA0* or *pA1*) – this puts the safe on standby – and then the correct button for lock B must be pushed (using action *pB0* or *pB1*). Any button pushing other than this sequence sets off an alarm, and the game is lost. Instead of having combinations written on a pieces of paper, we assume that there are two binary sensing actions, *sA* and *sB*, that cause the agent performing them to come to know the combination of the lock in question. We also assume that the agents act synchronously and in turn: P acts first and then they alternate. The goal in all cases will be to open the safe in exactly 4 steps without activating the alarm. We consider four variants of this setup with different assumptions.

Example 1: Suppose nothing is specified about the agents knowledge about the correct combinations of the locks. Actions are restricted in such a way that P and Q can only choose among actions $\{pA0, pA1, sA\}$ and $\{pB0, pB1, sB\}$, respectively. The actions are public so each agent gets to see the actions of the other agent. For this example, we want to say the agents can jointly open the safe. If the agents know nothing, the intuitive joint plan would look like this: P senses the combination of A, Q senses the combination of B, P pushes the correct button for A, and Q pushes the correct button for B. Note that if agents have extra information, e.g. P knows in advance the combination of A, other successful joint plans will exist, but as we will see, they do not cause any coordination problem.

Example 2: Suppose everything is exactly as in example 1 except that *sA* does not provide any information (the sensor is broken). In this case, we want to say that there is not enough information to conclude that the agents can open the safe. In fact, we will show that if P does not know the combination of lock A, they are provably not jointly able to open the safe.

Example 3: Suppose everything is as in example 1 except that the actions are not public, so the agents do not see what actions the other agent has performed (each just knows that some action has been performed by the other, so there is no confusion about whose turn it is). We will show that in this case seeing the other agent's actions is not necessary and that the agents are jointly able to open the safe, again without any need for extra assumptions about the beliefs of the agents.

Example 4: Suppose everything is as in example 3 except all actions are available to both agents, i.e., both P and Q can choose among actions $\{pA0, pA1, sA, pB0, pB1, sB\}$. We still assume that actions are private. As in example 2, we will show that there is not enough information to conclude the agents can open the safe. However, this is not because there is no good joint plan, instead the problem is that one agent might have extra knowledge which enables him to also open the safe on his own (if the other agent does not interfere) and since actions are private this can cause lack of coordination.

3 The formal framework

The basis of our framework for joint ability is the situation calculus [McCarthy and Hayes, 1969; Levesque *et al.*, 1998]. The situation calculus is a predicate calculus language for representing dynamically changing domains. A *situation* represents a possible state of the domain. There is a set of initial situations corresponding to the ways the domain might be initially. The actual initial state of the domain is represented by the distinguished initial situation constant, S_0. The term $do(a, s)$ denotes the unique situation that results from an agent doing action a in situation s. We use $do(\langle a_1, \cdots, a_n \rangle, s)$ as a shorthand for $do(a_n, do(\cdots, do(a1, s)) \cdots)$. Initial situations are defined as those that do not have a predecessor: $Init(s) \doteq \neg\exists a \exists s'.\ s = do(a, s')$. In general, the situations can be structured into a set of trees, where the root of each tree is an initial situation and the arcs are actions. The formula $s \sqsubseteq s'$ is used to state that there is a path from situation s to situation s'. Our account of joint ability will require some second-order features of the situation calculus, including quantifying over certain functions from situations to actions, that we call *strategies*.

Predicates and functions whose values may change from situation to situation (and whose last argument is a situation) are called *fluents*. The effects of actions on fluents are defined using successor state axioms [Reiter, 2001], which provide a succinct representation for both effect and frame axioms [McCarthy and Hayes, 1969]. To axiomatize a dynamic domain in the situation calculus, we use Reiter's [2001] action theory, which consists of (1) successor state axioms; (2) initial state axioms, describing the initial states of the domain including the initial beliefs of the agents; (3) precondition axioms, specifying the conditions under which each action can be executed; (4) unique names axioms for the actions, and (5) domain-independent foundational axioms (we adopt the ones given in [Levesque *et al.*, 1998] which accommodate multiple initial situations, but we do not describe them further here).

For our examples, we need eight fluents. The fluents *cA* and *cB* indicate the combination of locks A and B (i.e. true corresponds to button 1, and false corresponds to button 0 as the correct buttons that need to be pushed). The fluents *open*, *standby*, and *alarm* indicate whether the safe is open, the safe is on standby, and the alarm is activated, respectively. The fluent *time* indicates how many actions have been performed. Finally, the fluent *turn* is used to indicate whose turn it is to act, and the fluent B deals with the beliefs of the agents.

Moore [1985] defined a possible-worlds semantics for a logic of knowledge in the situation calculus by treating situations as possible worlds. Scherl and Levesque [2003] adapted this to Reiter's theory of action and gave a successor state axiom for B that states how actions, including sensing actions, affect knowledge. Shapiro *et al.* [1998] adapted this to handle the beliefs of multiple agents, and we adopt their account here. $B(x, s', s)$ will be used to denote that in situation s, agent x thinks that situation s' might be the actual situation. Note that the order of the situation arguments is reversed from the convention in modal logic for accessibility relations. Belief is then defined as an abbreviation:[1]

$$\mathit{Bel}(x, \phi[\mathit{now}], s) \ \doteq\ \forall s'.\ B(x, s', s) \supset \phi[s'].$$

We will also use the following abbreviation:

$$\mathit{BW}(x, \phi, s) \doteq \mathit{Bel}(x, \phi, s) \vee \mathit{Bel}(x, \neg\phi, s).$$

Mutual beliefs among the agents, denoted by *MBel*, can be defined either as a fix-point or by introducing a new accessibility relation using a second-order definition.

Our examples use the following successor state axioms:

- The combinations of the locks do not change over time: $\mathit{cA}(do(a, s)) \equiv \mathit{cA}(s)$, and $\mathit{cB}(do(a, s)) \equiv \mathit{cB}(s)$.
- If the safe is on standby and the alarm is not active, pushing the correct button for lock B opens the safe:
  $\mathit{open}(do(a, s)) \equiv \mathit{standby}(s) \wedge \neg \mathit{alarm}(s) \wedge$
  $\quad [\mathit{cB}(s) \wedge a = \mathit{pB1} \ \vee\ \neg \mathit{cB}(s) \wedge a = \mathit{pB0}] \vee \mathit{open}(s)$.
- If the alarm is not active, pushing the correct button for lock A puts the safe on standby:
  $\mathit{standby}(do(a, s)) \equiv \neg \mathit{alarm}(s) \wedge$
  $\quad [\mathit{cA}(s) \wedge a = \mathit{pA1} \vee \neg \mathit{cA}(s) \wedge a = \mathit{pA0}] \vee \mathit{standby}(s)$.
- The alarm is activated by pushing the wrong button, pushing a button of A if the safe is already on standby, or pushing any button if the safe is already open:
  $\mathit{alarm}(do(a, s)) \equiv \mathit{alarm}(s) \ \vee$
  $\quad \mathit{cA}(s) \wedge a = \mathit{pA0} \ \vee\ \neg \mathit{cA}(s) \wedge a = \mathit{pA1} \ \vee$
  $\quad \mathit{cB}(s) \wedge a = \mathit{pB0} \ \vee\ \neg \mathit{cB}(s) \wedge a = \mathit{pB1} \ \vee$
  $\quad \mathit{standby}(s) \wedge (a = \mathit{pA0} \ \vee\ a = \mathit{pA1}) \ \vee$
  $\quad \mathit{open}(s) \wedge [a = \mathit{pA0} \vee a = \mathit{pA1} \vee a = \mathit{pB0} \vee a = \mathit{pB1}]$.
- Belief changes due to sensing and other actions. We use the following type of successor state axiom proposed by Scherl and Levesque in the case where actions are public (see Section 4.3 for the case where actions are private):
  $B(x, s', do(a, s)) \equiv \exists s''.\ B(x, s'', s) \wedge s' = do(a, s'')$
  $\quad \wedge\ [\mathit{agent}(a) = x \supset (\mathit{SF}(a, s'') \equiv \mathit{SF}(a, s))]$.
  $\mathit{SF}(a, s) \ \equiv\ [a = \mathit{sA} \supset \mathit{cA}(s)] \ \wedge\ [a = \mathit{sB} \supset \mathit{cB}(s)]$.
  Thus when any action occurs, all agents learn that it has occurred. Moreover, when an agent performs *sA* or *sB*, he alone learns the corresponding lock combination.
- Each action uses one time step:
  $\mathit{time}(do(a, s)) = \mathit{time}(s) + 1$.
- Whose turn it is to act alternates between P and Q:
  $\mathit{turn}(do(a, s)) = x \ \equiv$
  $\quad \mathit{turn}(s) = Q \supset x = P \ \wedge\ \mathit{turn}(s) = P \supset x = Q$.

The examples also include the following initial state axioms:

- $\mathit{Init}(s) \supset \mathit{turn}(s) = P$. So, agent P gets to act first.
- In all initial situations, time starts at 0, the alarm is not active, the safe is not on standby and not open:
  $\mathit{Init}(s) \supset$
  $\quad \mathit{time}(s) = 0 \wedge \neg \mathit{alarm}(s) \wedge \neg \mathit{standby}(s) \wedge \neg \mathit{open}(s)$.
- Each agent initially knows that it is in an initial situation: $\mathit{Init}(s) \wedge B(x, s', s) \supset \mathit{Init}(s')$.
- B models knowledge, and hence beliefs must be true: $\mathit{Init}(s) \supset B(x, s, s)$.
- Each agent initially has introspection of her beliefs: $\mathit{Init}(s) \wedge B(x, s', s) \supset [\forall s''.\ B(x, s'', s') \equiv B(x, s'', s)]$.

The last two properties of belief can be shown to hold for all situations using the successor state axiom for B so that belief satisfies the modal system *KT45* [Chellas, 1980]. Since the axioms above are universally quantified, they are known to all agents, and in fact are common knowledge. We will let Σ denote the action theory containing the successor and initial state axioms above. All the examples in Section 4 will use Σ (with variations in the B or *SF* axiom) and in some cases with additional conditions about the beliefs of agents.

3.1 Our definition of joint ability

In this paper, for simplicity, we use [Ghaderi *et al.*, 2007]'s definition restricted to two agents (for the general definition see [Ghaderi *et al.*, 2007]). All of the definitions below are abbreviations for formulas in the language of the situation calculus presented above. The joint ability of two agents P and Q to achieve ϕ is defined as follows:

- P and Q can jointly achieve ϕ starting from s iff all combinations of their preferred strategies work together:
  $\mathit{JCan}(\phi, s) \doteq \forall \sigma_p, \sigma_q.\ \mathit{Pref}(P, \sigma_p, \phi, s) \wedge$
  $\quad \mathit{Pref}(Q, \sigma_q, \phi, s) \supset \mathit{Works}(\sigma_p, \sigma_q, \phi, s)$.
- The pair of strategies σ_p and σ_q works if there is a future situation where ϕ holds and the strategies prescribe the actions to get there according to whose turn it is:
  $\mathit{Works}(\sigma_p, \sigma_q, \phi, s) \ \doteq$
  $\quad \exists s''.\ s \sqsubseteq s'' \wedge \phi[s''] \wedge \forall s'.\ s \sqsubseteq s' \sqsubset s'' \supset$
  $\quad (\mathit{turn}(s') = P \supset do(\sigma_p(s'), s') \sqsubseteq s'') \wedge$
  $\quad (\mathit{turn}(s') = Q \supset do(\sigma_q(s'), s') \sqsubseteq s'')$.
- Agent x prefers strategy σ_x if it is kept for all levels n: $\mathit{Pref}(x, \sigma_x, \phi, s) \ \doteq\ \forall n.\ \mathit{Keep}(x, n, \sigma_x, \phi, s)$.
- *Keep* is defined inductively:[2]
  - At level 0, each agent keeps all of her strategies: $\mathit{Keep}(x, 0, \sigma_x, \phi, s) \ \doteq\ \mathit{Strategy}(x, \sigma_x)$.

[1] Free variables are assumed to be universally quantified from outside. If ϕ is a formula with a single free situation variable, $\phi[t]$ denotes ϕ with that variable replaced by situation term t. Instead of $\phi[\mathit{now}]$ we occasionally omit the situation argument completely.

[2] Strictly speaking, the definition we propose here is ill-formed. We want to use it with the second argument universally quantified (as in *Pref*). *Keep* and *GTE* actually need to be defined using second-order logic, from which the definitions here emerge as consequences. We omit the details for space reasons.

  - at level $n+1$, agent x keeps strategy σ_x if it was kept at level n and there was *not* a *better* kept σ'_x (σ'_x is better than σ_x if σ'_x is as good as, i.e. greater than or equal to, σ_x while σ_x is not as good as it):
    $\mathit{Keep}(x, n+1, \sigma_x, \phi, s) \doteq \mathit{Keep}(x, n, \sigma_x, \phi, s) \wedge$
    $\neg\exists\sigma'_x.\ \mathit{Keep}(x, n, \sigma'_x, \phi, s) \wedge$
    $\mathit{GTE}(x, n, \sigma'_x, \sigma_x, \phi, s) \wedge \neg\mathit{GTE}(x, n, \sigma_x, \sigma'_x, \phi, s)$.

- Strategy σ_x is as good as (Greater Than or Equal to) σ'_x for agent x at level n if x believes that whenever σ'_x works with strategies kept by the other agent y, so does σ_x. Note that here $x = P \wedge y = Q$ or $x = Q \wedge y = P$:
  $\mathit{GTE}(x, n, \sigma_x, \sigma'_x, \phi, s) \doteq$
  $\forall\sigma_y.\ \mathit{Bel}(x, [\mathit{Keep}(y, n, \sigma_y, \phi, \mathit{now}) \wedge$
  $\mathit{Works}(\sigma'_x, \sigma_y, \phi, \mathit{now}) \supset \mathit{Works}(\sigma_x, \sigma_y, \phi, \mathit{now})], s)$.

- Finally, strategies for an agent are functions from situations to actions such that the required action is legal and known to the agent whenever it is the agent's turn to act:
  $\mathit{Strategy}(x, \sigma) \doteq \forall s.\ \mathit{turn}(s) \neq x \supset \sigma(s) = \mathit{nil} \wedge$
  $\mathit{turn}(s) = x \supset \exists a. \mathit{Bel}(x, \sigma(\mathit{now}) = a, s) \wedge \mathit{Legal}(a)$.
  Legal will depend on the domain. For examples 1, 2 and 3, it is defined such that P can only do actions *pA0*, *pA1*, and *sA*, while Q can only do *pB0*, *pB1*, and *sB*. For example 4, all actions will be possible for both agents.

These formulas define joint ability in a way that resembles the iterative elimination of weakly dominated strategies of game theory [Osborne and Rubinstein, 1999]. As we will see in the examples next, the mere *existence* of a working strategy profile is not enough; the definition requires coordination among the agents in that *all* preferred strategies must work together.

4 Formalizing the Examples

In this section, we prove results about the four examples mentioned earlier. Due to lack of space we present only brief proof sketches. Note that the goal in all examples is to open the safe in exactly 4 steps without activating the alarm, i.e.

$$\phi(s) \doteq \mathit{open}(s) \wedge \neg\mathit{alarm}(s) \wedge \mathit{time}(s) = 4.$$

4.1 Example 1

Recall that for this example actions are divided between agents and are public. We show that the agents are jointly able to achieve the goal (and have mutual belief about this):

Theorem 1 $\Sigma \models \mathit{Init}(s) \supset \mathit{JCan}(\phi, s)$.

Actually, it is sufficient to show that the following holds:

Theorem 2 $\Sigma \models \mathit{Init}(s) \wedge \mathit{Keep}(P, 2, \sigma_p, \phi, s) \wedge$
$\mathit{Keep}(Q, 2, \sigma_q, \phi, s) \supset \mathit{Works}(\sigma_p, \sigma_q, \phi, s)$.

The proof is involved, so we just sketch the steps. Assume M is a model of Σ and μ a variable assignment such that $M, \mu \models \mathit{Init}(s) \wedge \mathit{Keep}(P, 2, \sigma_p, \phi, s) \wedge \mathit{Keep}(Q, 2, \sigma_q, \phi, s)$. We need to show that $\mathit{Works}(\sigma_p, \sigma_q, \phi, s)$ holds. Let $\overline{\sigma_p}$ and $\overline{\sigma_q}$ be strategies prescribing that P and Q initially sense the combination of locks A and B, respectively, and then push the correct button of the corresponding lock, in turn, i.e.:[3]

- $M, \mu \models \forall s, a1, a2, s'.\ \mathit{Init}(s) \supset$
  $\overline{\sigma_p}(s) = \mathit{sA} \wedge \overline{\sigma_p}(\mathit{do}(a1, s)) = \mathit{nil} \wedge$
  $\overline{\sigma_p}(\mathit{do}(\langle a1, a2\rangle, s)) = \mathit{pA}(s) \wedge$
  $[\mathit{do}(\langle a1, a2\rangle, s) \sqsubset s' \supset$
  $\mathit{turn}(s') = P \supset \overline{\sigma_p}(s') = \mathit{sA} \wedge$
  $\mathit{turn}(s') \neq P \supset \overline{\sigma_p}(s') = \mathit{nil}]$.
- $M, \mu \models \forall s, a1, a2, a3, s'.\ \mathit{Init}(s) \supset$
  $\overline{\sigma_q}(s) = \mathit{nil} \wedge \overline{\sigma_q}(\mathit{do}(a1, s)) = \mathit{sB} \wedge$
  $\overline{\sigma_q}(\mathit{do}(\langle a1, a2\rangle, s)) = \mathit{nil} \wedge$
  $\overline{\sigma_q}(\mathit{do}(\langle a1, a2, a3\rangle, s)) = \mathit{pB}(s) \wedge$
  $[\mathit{do}(\langle a1, a2, a3\rangle, s) \sqsubset s' \supset$
  $\mathit{turn}(s') = Q \supset \overline{\sigma_q}(s') = \mathit{sB} \wedge$
  $\mathit{turn}(s') \neq Q \supset \overline{\sigma_q}(s') = \mathit{nil}]$.

It can be easily shown that functions $\overline{\sigma_p}$ and $\overline{\sigma_q}$ are in fact strategies for P and Q that together achieve the goal in *all* initial situations, and hence each survives the first round of elimination for the corresponding agent. Also, after the first round of eliminations, the following holds *at level 1*:

Theorem 3 *$\overline{\sigma_p}$ and $\overline{\sigma_q}$ are as good as any other strategies:*

- $M, \mu \models \forall s, \sigma_p.\ \mathit{Init}(s) \wedge \mathit{Strategy}(P, \sigma_p) \supset$
  $\mathit{GTE}(P, 1, \overline{\sigma_p}, \sigma_p, \phi, s)$.
- $M, \mu \models \forall s, \sigma_q.\ \mathit{Init}(s) \wedge \mathit{Strategy}(Q, \sigma_q) \supset$
  $\mathit{GTE}(Q, 1, \overline{\sigma_q}, \sigma_q, \phi, s)$.

By theorem 3, $\mathit{GTE}(P, 1, \overline{\sigma_p}, \sigma_p, \phi, s)$ holds, and by assumption $\mathit{Keep}(P, 2, \sigma_p, \phi, s)$ holds, therefore we must have $\mathit{GTE}(P, 1, \sigma_p, \overline{\sigma_p}, \phi, s)$. Then, since in all initial situations $\mathit{Works}(\overline{\sigma_p}, \overline{\sigma_q}, \phi, s)$ holds, we must have $\mathit{Works}(\sigma_p, \overline{\sigma_q}, \phi, s)$. By a similar argument, we have $\mathit{GTE}(Q, 1, \sigma_q, \overline{\sigma_q}, \phi, s)$. This together with $\mathit{Works}(\sigma_p, \overline{\sigma_q}, \phi, s)$ obtained above, leads to $\mathit{Works}(\sigma_p, \sigma_q, \phi, s)$ as desired and thus Theorem 2 holds. ■

Theorem 3 itself can proved by the following two lemmas:

Lemma 1 *For any strategy for P that survives the first elimination round, if its first action is to push the correct A button, the action prescribed in response to Q doing sB must be sA:*
$\Sigma \models \mathit{Init}(s) \wedge \mathit{Keep}(P, 1, \sigma_p, \phi, s) \wedge \sigma_p(s) = \mathit{pA}(s) \supset$
$\sigma_p(\mathit{do}(\langle \mathit{pA}(s), \mathit{sB}\rangle, s)) = \mathit{sA}$.

Lemma 2 *For any strategy for Q that survives the first round of elimination, its first action in response to P doing sA must be doing sB, and then if P continues by pushing the correct button of lock A, Q must push the correct button of B:*
$\Sigma \models \mathit{Init}(s) \wedge \mathit{Keep}(Q, 1, \sigma_q, \phi, s) \supset$
$\sigma_q(\mathit{do}(\mathit{sA}, s) = \mathit{sB} \wedge \sigma_q(\mathit{do}(\langle \mathit{sA}, \mathit{sB}, \mathit{pA}(s)\rangle, s)) = \mathit{pB}(s)$.

The proofs of lemmas 1 and 2 are omitted but they use the fact that in a given initial situation s there are only 3 legal sequences of length 4 that can open the safe without activating the alarm: $[\mathit{sA}; \mathit{sB}; \mathit{pA}(s); \mathit{pB}(s)]$, $[\mathit{pA}(s); \mathit{sB}; \mathit{sA}; \mathit{pB}(s)]$, and $[\mathit{pA}(s); \mathit{pB}(s); \mathit{sA}; \mathit{sB}]$, where $\mathit{pA}(s)$ and $\mathit{pB}(s)$ correspond to the correct push actions for lock A and B in s, respectively.

We remind the reader that the reason that the above proofs are involved is that we have not specified anything about the beliefs of agents about the locks combinations and/or each other. Our theorems hold no matter what beliefs the agents have about this (e.g. if P and Q know the combination of both locks but neither knows what the other agent knows, they can still coordinate to open the safe despite the existence of many working plans). See example 4 as a case where the existence of multiple joint plans can cause lack of coordination.

[3] In what follows, we use $\mathit{pA}(s)$ as a shorthand for the *correct push action* for lock A in situation s. Any formula ψ that mentions $\mathit{pA}(s)$ with free variable s stands for $(\mathit{cA}(s) \supset \psi[\mathit{pA}(s)/\mathit{pA1}]) \wedge (\neg\mathit{cA}(s) \supset \psi[\mathit{pA}(s)/\mathit{pA0}])$, where $\psi[u/v]$ is replacing all free occurrences of u by v in ψ. We use a similar definition for $\mathit{pB}(s)$.

4.2 Example 2

In this example, action *sA* does not provide new information. To handle this, let Σ_2 be exactly like Σ except the *SF* axiom is replaced by $\mathit{SF}(a, s) \equiv [a = \mathit{sB} \supset \mathit{cB}(s)]$. The information in Σ_2 is not enough to conclude joint ability. In fact, we show that if P does not know the combination of lock A they *cannot* open the safe (even if Q knows both combinations):

Theorem 4 $\Sigma_2 \models \mathit{Init}(s) \wedge \neg \mathit{BW}(P, \mathit{cA}, s) \supset \neg \mathit{JCan}(\phi, s)$.

Proof sketch: Let M be a model of Σ_2 and μ be a variable assignment such that $M, \mu \models \mathit{Init}(s) \wedge \neg \mathit{BW}(P, \mathit{cA}, s)$, it is sufficient to show that there exists a pair of *preferred* strategies for P and Q that does not achieve the goal. Since P does not know the combination of A, there is at least another accessible initial situation s' such that $\mathit{cA}(s) \equiv \neg \mathit{cA}(s')$. Note that the function $\overline{\sigma_p}$ defined in example 1 is not a strategy in this model as P now does not know the combination of lock A even after performing *sA*. We can show that P has at least two preferred strategies σ_p and σ'_p such that $M, \mu \models \sigma_p(s) = \mathit{pA0} \ \wedge \ \sigma'_p(s) = \mathit{pA1}$. However, for *any* strategy σ_q for Q, one of the pair (σ_p, σ_q) or (σ'_p, σ_q) does not work in s, as the first action by P activates the alarm. ■

4.3 Example 3

In this example, everything is the same as in example 1 except that actions are now private, so the other agent does not see what actions are performed by the other agent (but each agent is aware of her own actions including the sensing results if any). To accommodate for this, we define Σ_3 exactly as Σ except we modify the successor state axiom for B as follows:

$B(x, s', do(a, s)) \equiv \exists s'', a''.$
$\quad B(x, s'', s) \wedge s' = do(a'', s'') \wedge \mathit{Legal}(a'') \wedge$
$\quad [\mathit{agent}(a) = x \ \supset \ a = a'' \wedge (\mathit{SF}(a, s'') \equiv \mathit{SF}(a, s))]$.

Despite actions being private, we can prove that the agents have joint ability to open the safe (again without any need for additional specifications about their beliefs):

Theorem 5 $\Sigma_3 \models \mathit{Init}(s) \supset \mathit{JCan}(\phi, s)$.

The proof is similar to that of example 1. Note that $\overline{\sigma_p}$ and $\overline{\sigma_q}$ used there to eliminate non-promising strategies did not rely on actions of the other agent and are applicable here as well. However, the proof for Theorem 3 is slightly different.

4.4 Example 4

In this example, all actions are legal for both agents but, as in example 3, are private. To handle this, let Σ_4 be like Σ_3 except that *Legal* is defined such that both agents can perform any of actions *pA0*, *pB0*, *pA1*, *pB1*, *sA*, and *sB*.[4] Under these assumptions we cannot conclude that the agents have joint ability to open the safe; in fact we show that if it is initially mutually known that P does not know the lock combinations and Q knows both combinations they *cannot* open the safe:

Theorem 6 $\Sigma_4 \models \mathit{Init}(s) \wedge \mathit{MBel}(\mathit{BW}(Q, \mathit{cA}) \wedge \mathit{BW}(Q, \mathit{cB}) \wedge \neg \mathit{BW}(P, \mathit{cA}) \wedge \neg \mathit{BW}(P, \mathit{cB}), s) \supset \neg \mathit{JCan}(\phi, s)$.

	$[sB;pB] \rightarrow \sigma_q^1$	$\sigma_q^2 \leftarrow [pA;pB]$	σ_q
$[sA;pA] \longrightarrow \sigma_p^1$	✓	X	
$[sA;sA] \longrightarrow \sigma_p^2$	X	✓	
σ_p			

Figure 1: For private shared actions, if it is mutually believed that Q knows the locks combinations and P does not, multiple *incompatible preferred* plans exist that cause lack of coordination. In the above matrix, a ✓ at i, j corresponds to $\mathit{MBel}(\mathit{Works}(i, j, \phi), s)$ and an X corresponds to $\mathit{MBel}(\neg \mathit{Works}(i, j, \phi), s)$.

The interesting point here is that unlike in example 2, this is not because no joint plan exists. Quite the opposite, there are multiple joint plans that open the safe but the agents cannot coordinate (assuming no prior conventions can be relied upon). To see this consider any model M of Σ_4 and variable assignment μ where $M, \mu \models \mathit{Init}(s) \wedge \mathit{MBel}(\mathit{BW}(Q, \mathit{cA}) \wedge \mathit{BW}(Q, \mathit{cB}) \wedge \neg \mathit{BW}(P, \mathit{cA}) \wedge \neg \mathit{BW}(P, \mathit{cB}), s)$. We can show that P and Q each prefers at least two strategies whose combinations do not always work (i.e. there is lack of coordination). Let σ_p^1 be a strategy for P that prescribes sensing the combination of lock A and pushing its correct button as P's first and second (non-*nil*) actions (represented by [*sA*; *pA*]). Also, let σ_p^2 be a strategy for P that always prescribes performing *sA* whenever it is P's turn (represented by [*sA*; *sA*]). Similarly, let σ_q^1 be a strategy for Q that says to sense the combination of lock B and then to push its correct button as Q's first and second non-*nil* actions (represented by [*sB*; *pB*]). Finally, let σ_q^2 be a strategy for Q that prescribes pushing the correct button of lock A and B as Q's first and second non-*nil* actions (represented by [*pA*; *pB*]). Note that since Q knows both combinations, σ_q^2 is in fact a valid strategy. Clearly, we have $M, \mu \models \mathit{MBel}(\mathit{Works}(\sigma_p^1, \sigma_q^1, \phi) \wedge \mathit{Works}(\sigma_p^2, \sigma_q^2, \phi) \wedge \neg \mathit{Works}(\sigma_p^1, \sigma_q^2, \phi) \wedge \neg \mathit{Works}(\sigma_p^2, \sigma_q^1, \phi), s)$, see Fig. 1. To show that the agents are not able to open the safe, it remains to show that P and Q never eliminate these strategies:

Lemma 3 *P prefers σ_p^1 and σ_p^2. Q prefers σ_q^1 and σ_q^2:*

- $M, \mu \models \forall i.\ \mathit{Keep}(P, i, \sigma_p^1, \phi, s) \wedge \mathit{Keep}(P, i, \sigma_p^2, \phi, s)$.
- $M, \mu \models \forall i.\ \mathit{Keep}(Q, i, \sigma_q^1, \phi, s) \wedge \mathit{Keep}(Q, i, \sigma_q^2, \phi, s)$.

We sketch the proof for the 1st elimination round ($i = 1$), the generalization to all i's is done using simple induction. Assume to the contrary $M, \mu \models \neg \mathit{Keep}(P, 1, \sigma_p^1, \phi, s)$. Then there must exist a better strategy σ_p for P such that $\mathit{GTE}(P, 0, \sigma_p, \sigma_p^1, \phi, s)$ and $\neg \mathit{GTE}(P, 0, \sigma_p^1, \sigma_p, \phi, s)$. Hence, since $M, \mu \models \mathit{MBel}(\mathit{Works}(\sigma_p^1, \sigma_q^1, \phi), s)$, we must have $M, \mu \models \mathit{Bel}(P, \mathit{Works}(\sigma_p, \sigma_q^1, \phi), s)$. However, any strategy for P that works with σ_q^1 in *all* P's accessible initial situations must prescribe doing *sA* and then *pA* as P's first two non-*nil* actions, respectively.[5] Hence, the first two actions of σ_p and σ_p^1 are the same, which contradicts the assumption of σ_p being better than σ_p^1. Therefore, P keeps σ_p^1 at level 1. Similarly, we can show that if there were strategy σ_p better

[4] Technically, every action a has an agent parameter as its first argument where, e.g., $\mathit{agent}(\mathit{pA0}(x)) = x$. To simplify the presentation we have omitted the agent argument. Very minor modifications to the formulas presented here are needed to restore the argument.

[5] P does not know the combination of lock A, so there exist two accessible initial situations that differ on *cA*. Any strategy that prescribes first doing *pA0* (or *pA1*) activates the alarm in one of them.

than σ_p^2 then $M, \mu \models Bel(P, Works(\sigma_p, \sigma_q^2, \phi), s)$. However, any strategy σ_p that works with σ_q^2 in *all* P's accessible initial situations must prescribe doing nothing but sensing as P's first and second (non-*nil*) actions. It can then be shown that $M, \mu \models GTE(P, 0, \sigma_p^2, \sigma_p, \phi, s)$ which contradicts σ_p being better than σ_p^2. So, σ_p^2 is also kept at level 1. Finally, there are analogous arguments for Q keeping σ_q^1 and σ_q^2 at level 1. ∎

5 Discussion and Future Work

In this paper, we extended Ghaderi *et al.* [2007]'s account of joint ability to domains with *sensing actions*, actions that allow agents to acquire new information as they proceed. We proposed ways of modeling the effects of such sensing actions on the agents' knowledge in the account. In such settings, strategies branch on sensing outcomes (as well as on observed actions by others), and the number of strategies typically grows extremely large. We showed that the symbolic proof techniques proposed in [Ghaderi *et al.*, 2007] could be generalized to establish joint ability or lack of joint ability in domains with sensing actions, even with very incomplete specifications of the agents' knowledge.

Our account of ability generalizes previous work on single agent ability [Moore, 1985; Davis, 1994; Lespérance *et al.*, 2000]. We go beyond these single agent accounts by modeling how the knowledge of all the agents changes as they act and by ensuring that the team remains coordinated — all of the agents' preferred strategies must work together.

Also related is work on logics of games [Pauly, 2002; van der Hoek and Wooldridge, 2003]. As mentioned earlier, these frameworks are propositional, and thus less expressive than ours. Moreover, they ignore the need for coordination inside a coalition, which is only reasonable if the agents can communicate arbitrarily to agree on a joint strategy.

Our approach goes beyond classical game theory [Osborne and Rubinstein, 1999] in that we can reason about joint ability even in the presence of incomplete specifications of the structure of the game including the beliefs of the agents. See [Ghaderi *et al.*, 2007] for more discussion of the relationship between the two accounts.

In this paper, for simplicity, we used Ghaderi *et al.*'s formalization of joint ability restricted to teams of two agents; see [Ghaderi *et al.*, 2007] for the general multiagent version. Their paper also discusses how agents that are outside of the team can be handled, i.e. by ensuring that the team's strategies achieve the goal for all of the outside agents' strategies.

As mentioned earlier, our approach can also handle *informing* communication actions where the truth value of a proposition or the value of a fluent is communicated by an agent to one or several other agents. It is straightforward to reformulate the examples considered in this paper to involve communication actions; instead of simply sensing a lock combination, an agent asks another "informer" agent for its value.

An issue for future work is examining how different ways of comparing strategies (the *GTE* order) affect the notion of joint ability. With the current *GTE* order, each agent compares her strategies by examining how they work when paired with the strategies of the other agent *in each accessible situation separately*. Another possibility is that, for example, each agent performs the comparison based on whether she *believes* her strategies work with those of the other agent (i.e. *Bel* is distributed over the implication in the *GTE* definition). Both definitions give the right results for our examples and others.

Also, in future work, we would like to generalize *Legal*/*Poss* to be situation dependent, and devise ways of handling *conventions*, i.e. mutually believed rules that allow agents to stay coordinated. It would also be good to explore how the framework can be used in automated verification and in multiagent planning.

References

[Chellas, 1980] B. Chellas. *Modal logic: an introduction*. United Kingdom: Cambridge University Press, 1980.

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

[Ghaderi *et al.*, 2007] Hojjat Ghaderi, Hector Levesque, and Yves Lespérance. A logical theory of coordination and joint ability. In *AAAI'07*, pages 421–426, 2007.

[Lespérance *et al.*, 2000] Yves Lespérance, Hector J. Levesque, Fangzhen Lin, and Richard B. Scherl. Ability and knowing how in the situation calculus. *Studia Logica*, 66(1):165–186, 2000.

[Levesque *et al.*, 1998] H. Levesque, F. Pirri, and R. Reiter. Foundations for the situation calculus. *Electronic Transactions on Artificial Intelligence*, 2(3-4):159–178, 1998.

[McCarthy and Hayes, 1969] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. 1969.

[Moore, 1985] R. Moore. A formal theory of knowledge and action. In J. Hobbs and R. Moore, editors, *Formal Theories of the Commonsense World*, pages 319–358, 1985.

[Osborne and Rubinstein, 1999] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. MIT Press, 1999.

[Pauly, 2002] M. Pauly. A modal logic for coalitional power in games. *Logic and Computation*, 12(1):149–166, 2002.

[Reiter, 2001] Raymond Reiter. *Knowledge in Action: Logical Foundations for Specifying & Implementing Dynamical Systems*. The MIT Press, 2001.

[Scherl and Levesque, 2003] R. Scherl and H. Levesque. Knowledge, action, and the frame problem. *Artificial Intelligence*, 144:1–39, 2003.

[Shapiro *et al.*, 1998] S. Shapiro, Y. Lespérance, and H. Levesque. Specifying communicative multi-agent systems. In W. Wobcke, M. Pagnucco, and C. Zhang, editors, *Agents and Multiagent systems*, pages 1–14. Springer-Verlag, 1998.

[van der Hoek and Wooldridge, 2003] W. van der Hoek and M. Wooldridge. Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications. *Studia Logica*, 75(1):125–157, 2003.

[Wooldridge and Jennings, 1999] M. Wooldridge and N. R. Jennings. The cooperative problem-solving process. *Journal of Logic and Computation*, 9(4):563–592, 1999.

Definability and Process Ontologies

Michael Grüninger
Department of Mechanical and Industrial Engineering
University of Toronto
Toronto, Ontario, Canada
gruninger@mie.utoronto.ca

Abstract

In this paper, we use the notions of relative interpretations and definable models from mathematical logic to compare different ontologies and also to evaluate the limitations of particular ontologies. In particular, we characterize the relationship between the theories within the first-order PSL Ontology and two other ontologies – a first-order theory of time and Reiter's second-order axiomatization of situation calculus.

1 Introduction

Representing activities and the constraints on their occurrences is an integral aspect of commonsense reasoning, particularly in manufacturing, enterprise modelling, and autonomous agents or robots. There have been a variety of process ontologies developed within the artificial intelligence community, particularly in the context of robotics and planning systems.

In this paper, we use the notions of relative interpretations and definable models to compare different process ontologies and also to evaluate the limitations of particular ontologies. In particular, we characterize the relationship between the theories within the first-order PSL Ontology and two other ontologies – a first-order theory of time and Reiter's second-order axiomatization of situation calculus. There are two major kinds of results – relative interpretation theorems (which show the conditions under which two ontologies are equivalent), and nondefinability theorems (which show that one ontology is in some sense stronger since it is able to define concepts that other ontologies cannot define).

2 Relationships between Theories

Different ontologies within the same language can be compared using the notions of satisfiability, entailment, and independence. More difficult is to compare ontologies that are axiomatized in different languages; in such cases, we need to determine whether or not the lexicon of one ontology can be interpreted in the lexicon of the other ontology. In this section, we review the basic concepts from model theory that will supply us with the techniques for comparing ontologies in different languages.

2.1 Relative Interpretations of Theories

We will adopt the following definition from (Enderton 1972):

Definition 1 *An interpretation π of a theory T_0 with language L_0 into a theory T_1 with language L_1 is a function on the set of parameters of L_0 such that*

1. *π assigns to $\forall$ a formula $\pi_\forall$ of L_1 in which at most v_1 occurs free, such that*

$$T_1 \models (\exists v_1)\, \pi_\forall$$

2. *π assigns to each n-place relation symbol P a formula π_P of L_1 in which at most the variables $v_1, ..., v_n$ occur free.*
3. *π assigns to each n-place function symbol f a formula π_f of L_1 in which at most the variables $v_1, ..., v_n, v_{n+1}$ occur free, such that*

$$T_1 \models (\forall v_1, ..., v_n)\, \pi_\forall(v_1) \wedge ... \wedge \pi_\forall(v_n)$$
$$\supset (\exists x)(\pi_\forall(x) \wedge ((\forall v_{n+1})(\pi_f(v_1, ..., v_{n+1}) \equiv (v_{n+1} = x))))$$

4. *For any sentence σ in L_0,*

$$T_0 \models \sigma \Rightarrow T_1 \models \pi(\sigma)$$

2.2 Definable Interpretations

Relative interpretations specify mappings between theories; we are also interested in specifying mappings between models of the theories. Such an approach will also provide with a means of proving that no relative interpretation exists between two particular theories.

We begin with the notion of definable sets within a structure.

Definition 2 *Let $\mathcal{M}$ be a structure in a language L.*

A set $X \subseteq M^n$ is definable in $\mathcal{M}$ iff there is a formula $\varphi(v_1, ..., v_n, w_1, ..., w_m)$ of L and $\overline{\mathbf{b}} \in M^m$ such that

$$X = \{\overline{\mathbf{a}} \in M^n \,:\, \mathcal{M} \models \varphi(\overline{a}, \overline{b})\}$$

X is A-definable if there is a formula $\psi(\overline{v}, w_1, ..., w_l)$ and $\overline{\mathbf{b}} \in A^l$ such that

$$X = \{\overline{\mathbf{a}} \in M^n \,:\, \mathcal{M} \models \varphi(\overline{\mathbf{a}}, \overline{\mathbf{b}})\}$$

Using this definition, we can adopt the following approach from (Marker 2002):

Definition 3 *Let $\mathcal{N}$ be a structure in $\mathcal{L}_0$ and let $\mathcal{M}$ be a structure in $\mathcal{L}$. We say that $\mathcal{N}$ is definable in $\mathcal{M}$ iff we can find a definable subset X of M^n and we can interpret the symbols of $\mathcal{L}_0$ as definable subsets and functions on X so that the resulting structure in $\mathcal{L}_0$ is isomorphic to $\mathcal{N}$.*

The relationship between relative interpretations of theories and definable interpretations of structures is captured in a straightforward way by the following proposition:

Proposition 1 *If there exists an interpretation of T_1 into T_2, then every model of T_1 is definable in some model of T_2.*

Our primary tool for proving that the models of one ontology are not definable in the models of another ontology will be the following proposition from (Marker 2002):

Proposition 2 *Let $\mathcal{M}$ be a structure. If $X \subset M^n$ is A-definable, then every automorphism of $\mathcal{M}$ that fixes the set A pointwise fixes X setwise (that is, if σ is an automorphism of $\mathcal{M}$ and $\sigma(a) = a$ for all $a \in A$, then $\sigma(X) = X$).*

Using this proposition, we can show that a relation is not definable in some structure if there exists an automorphism of the structure that does not preserve the relation.

3 Definability and Time Ontologies

3.1 Linear Time with Endpoints

Consider the ontology $T_{linear-time}$[1] of linear time without endpoints (Hayes 1996). The countable models of this ontology are isomorphic to countably infinite linear orderings with no initial or final element.

Lemma 1 *Let $\mathcal{T}$ be a model of T_{linear_time} that is either discrete or dense.*

The set of automorphisms $Aut(\mathcal{T})$ does not fix any timepoints.

Proof: A model $\mathcal{T}$ of T_{linear_time} is discrete iff it contains an elementary subordering that isomorphic to $\mathbb{Z}$, and $Aut(\mathbb{Z})$ does not fix any elements of $\mathbb{Z}$.

A model $\mathcal{T}$ of T_{linear_time} is dense iff it contains an elementary subordering that is isomorphic to $\mathbb{Q}$, and $Aut(\mathbb{Q})$ does not fix any elements of $\mathbb{Q}$. □

In other words, for any timepoint in $\mathcal{T}$, there exists another timepoint which is the image of some automorphism of $\mathcal{T}$, whenever $\mathcal{T}$ is either discrete or dense.

3.2 Relationship to PSL-Core

The purpose of PSL-Core ((Gruninger 2004), (Bock & Gruninger 2005)) is to axiomatize a set of intuitive semantic primitives that are adequate for describing the fundamental concepts of manufacturing processes. Consequently, this characterization of basic processes makes few assumptions about their nature beyond what is needed for describing those processes, and it is therefore rather weak in terms of logical expressiveness.

Within PSL-Core [2], there are four kinds of entities required for reasoning about processes – activities, activity occurrences, timepoints, and objects. Activities may have multiple occurrences, or there may exist activities which do not occur at all. Timepoints are linearly ordered, forwards into the future, and backwards into the past. Finally, activity occurrences and objects are associated with unique timepoints that mark the begin and end of the occurrence or object.

Lemma 2 *A model of $T_{pslcore}$ in which the ordering over timepoints is either discrete or dense is not definable in any model of $T_{linear-time}$.*

Proof: Let $\mathcal{T}$ be a model of T_{linear_time} and let $\mathcal{M}$ be a model of $T_{pslcore}$ in which the ordering over timepoints is either discrete or dense.

By Lemma 1, the set of automorphisms $Aut(\mathcal{T})$ does not fix any timepoints, so that for any timepoint $\mathbf{t}$ there exists $\varphi \in Aut(\mathcal{T})$ such that $\varphi(\mathbf{t}) \neq \mathbf{t}$.

Since $\mathbf{beginof}$ is a function, activity occurrences have unique beginning timepoints, so that we have

$$\langle \mathbf{o}, \mathbf{t} \rangle \in \mathbf{beginof} \Rightarrow \langle \mathbf{o}, \varphi(\mathbf{t}) \rangle \notin \mathbf{beginof}$$

By Proposition 2, the $\mathbf{beginof}$ function is not definable in $\mathcal{T}$, and hence $\mathcal{M}$ is not definable in $\mathcal{T}$. □

Theorem 1 *There does not exist an interpretation of $T_{pslcore}$ into $T_{linear-time}$.*

Proof: This follows from Proposition 1 and Lemma 1. □

By Theorem 1, we cannot use a time ontology alone to reason about activities and their occurrences.

4 Definability and Situation Calculus

In this section, we characterize the relationship between Reiter's second-order axiomatization of the situation calculus and three core theories within the first-order PSL Ontology.

4.1 Axiomatization of Situation Calculus

Consider the theory $T_{sitcalc}$ which is Reiter's second-order axiomatization of the situation calculus ((Reiter 2001), (Levesque *et al.* 1997)). Let $T_{sittime}$ be Pinto's axiomatization of time for situation trees ((Pinto & Reiter 1995)) and let $T_{sitfluent}$ be Pinto's axiomatization of the $holds$ relation[3].

4.2 Relationship to PSL-Core

Theorem 2 *There exists an interpretation of $T_{pslcore}$ into $T_{sitcalc} \cup T_{sittime}$.*

Proof: Suppose

$$\pi_{\mathbf{occurrence\_of}}(s, a) = ((\exists s_1)\ s = do(a, s_1))$$

$$\pi_{\mathbf{activity}}(a) = ((\exists s_1, s_2)\ s = do(a, s_1))$$

$$\pi_{\mathbf{activity\_occurrence}}(s) = ((\exists a, s_1)\ s = do(a, s_1))$$

$$\pi_{\mathbf{timepoint}}(t) = ((\exists s)\ (start(s) = t))$$

[1] The axioms for $T_{linear-time}$ in CLIF (Common Logic Interchange Format) can be found at `http://www.stl.mie.utoronto.ca/colore/linear-time.clif`

[2] The axiomatization of PSL-Core (also referred to as $T_{pslcore}$) in CLIF can be found at `http://www.mel.nist.gov/psl/psl-ontology/psl_core.html`

[3] The axioms of $T_{sitcalc}$ in CLIF can be found at `http://stl.mie.utoronto.ca/colore/sitcalc.clif`.

The axioms of $T_{sittime}$ can be found at `http://stl.mie.utoronto.ca/colore/sittime.clif`. The axioms of $T_{sitfluent}$ can be found at `http://stl.mie.utoronto.ca/colore/sitfluent.clif`.

$$\pi_{\mathbf{beginof}}(s,t) = ((start(s) = t))$$
$$\pi_{\mathbf{endof}}(s,t) = ((\exists a)\ (end(s,a) = t))$$

It is straightforward to verify that these mappings and the axioms of $T_{sitcalc} \cup T_{sittime}$ entail the axioms of $T_{pslcore}$. □

Of course, it is not surprising to see that there exists an interpretation of $T_{pslcore}$ into $T_{sitcalc} \cup T_{sittime}$, since the theory $T_{pslcore}$ was designed to be the weakest process ontology that is shared by other process ontologies.

4.3 Relationship to Occurrence Trees

Within the PSL Ontology, the theory $T_{occtree}$ extends the theory of $T_{pslcore}$[4]. An occurrence tree is a partially ordered set of activity occurrences, such that for a given set of activities, all discrete sequences of their occurrences are branches of the tree.

An occurrence tree contains all occurrences of *all* activities; it is not simply the set of occurrences of a particular (possibly complex) activity. Because the tree is discrete, each activity occurrence in the tree has a unique successor occurrence of each activity. Every sequence of activity occurrences has an initial occurrence (which is the root of an occurrence tree).

Although occurrence trees characterize all sequences of activity occurrences, not all of these sequences will intuitively be physically possible within the domain. We therefore consider the subtree of the occurrence tree that consists only of *possible* sequences of activity occurrences; this subtree is referred to as the legal occurrence tree.

Occurrence trees are closely related to situation trees, which are the models of Reiter's axiomatization of situation calculus; the following theorems make this intuition more precise.

Theorem 3 *There exists an interpretation of* $T_{occtree} \cup T_{pslcore}$ *into* $T_{sitcalc} \cup T_{sittime}$.

Proof: Suppose

$$\pi_{\mathbf{earlier}}(s_1, s_2) = s_1 < s_2$$
$$\pi_{\mathbf{generator}}(a) = (\exists s_1, s_2)\ s = do(a, s_1)$$
$$\pi_{\mathbf{arboreal}}(s) = (\exists a, s_1)\ s = do(a, s_1)$$
$$\pi_{\mathbf{successor}}(a, s) = do(a, s)$$
$$\pi_{\mathbf{initial}}(s) = (s = do(a, S_0))$$
$$\pi_{\mathbf{legal}}(s) = (executable(s))$$

It is straightforward to verify that these mappings and the axioms of $T_{sitcalc} \cup T_{sittime}$ entail the axioms of $T_{occtree} \cup T_{pslcore}$. □

What of the converse direction – does there exist an interpretation of $T_{sitcalc} \cup T_{sittime}$ into $T_{occtree} \cup T_{pslcore}$. The primary difference between $T_{occtree}$ and $T_{sitcalc}$ is the existence of models of $T_{occtree}$ that are occurrence trees with branches that are not isomorphic to the standard models of the theory $Th(\mathbb{N}, 0, S, <)$; such trees cannot be isomorphic to situation trees.

Definition 4 *WFAS is the first-order axiom schema*

$$(\forall s)\ (\phi(s) \wedge arboreal(s))$$
$$\supset ((\exists x)\phi(x) \wedge earlier(x,s) \wedge ((\forall y) earlier(y,x) \supset \neg\phi(y)))$$

for any first-order formula $\phi(x)$.

This axiom schema is equivalent to saying that all first-order definable sets of elements in an occurrence tree are well-founded.

Theorem 4 *Let ACA be a sentence of the form*

$$(\forall a, s_1, s_2)\ (s_2 = do(a, s_1)) \supset (a = A_1) \vee ... \vee (a = A_n)$$

There exists an interpretation of $T_{sitcalc} \cup ACA$ *into* $T_{occtree} \cup T_{pslcore} \cup WFAS$.

Proof: (Sketch) Suppose

$$\pi_{<}(S_0, s_2) = (\exists s)\, initial(s) \wedge (earlier(s, s_2) \vee (s = s_2)$$
$$(s_1 \neq S_0) \Rightarrow \pi_{<}(s_1, s_2) = earlier(s_1, s_2)$$
$$\pi_{\mathbf{do}}(a, S_0) = (\exists s)\ initial(s) \wedge occurrence\_of(s, a)$$
$$(s_1 \neq S_0) \Rightarrow \pi_{\mathbf{do}}(a, s) = successor(a, s)$$
$$\pi_{\mathbf{executable}}(s) = (legal(s))$$

Since the interpretation of theories is specified with respect to first-order entailment, we only need to show that the first-order consequences are preserved by the interpretation.

The techniques introduced in (Doets 1989) and (Backofen, Rogers, & Vijay-Shanker 1995) can be used to show that the models of $T_{occtree} \cup T_{pslcore} \cup WFAS$ are elementarily equivalent to models of $T_{sitcalc} \cup ACA$. □

4.4 Relationship to Discrete States

Most applications of process ontologies are used to represent dynamic behaviour in the world so that intelligent agents may make predictions about the future and explanations about the past. In particular, these predictions and explanations are often concerned with the state of the world and how that state changes. The PSL core theory T_{disc_state} is intended to capture the basic intuitions about states and their relationship to activities[5].

Within the PSL Ontology, state is changed by the occurrence of activities. Intuitively, a change in state is captured by a state that is either achieved or falsified by an activity occurrence. Furthermore, state can only be changed by the occurrence of activities. Thus, if some state holds after an activity occurrence, but after an activity occurrence later along the branch it is false, then an activity must occur at some point between that changes the state. This also leads to the requirement that the state holding after an activity occurrence will be the same state holding prior to any immediately succeeding occurrence, since there cannot be an activity occurring between the two by definition.

Theorem 5 *There exists an interpretation of* $T_{disc\_state} \cup T_{occtree} \cup T_{pslcore}$ *into* $T_{sitcalc} \cup T_{sittime} \cup T_{sitfluent}$.

[4] The axioms of $T_{occtree}$ in CLIF can be found at `http://www.mel.nist.gov/psl/psl-ontology/part12/occtree.th.html`

[5] The axioms of T_{disc_state} in CLIF can be found at `http://www.mel.nist.gov/psl/psl-ontology/part12/disc_state.th.html`

Proof: Suppose

$$(s \neq S_0) \Rightarrow \pi_{\mathbf{holds}}(f, s) = holds(f, s)$$

$$\pi_{\mathbf{prior}}(f, s) = (((\forall s, s', a)\ s = do(a, s') \supset holds(f, s))$$
$$\wedge(((\exists s, s', a)\ s = do(a, s')) \vee holds(f, S_0)))$$

It is straightforward to verify that these mappings and the axioms of $T_{sitcalc} \cup T_{sittime} \cup T_{sitfluent}$ entail the axioms of $T_{disc\_state} \cup T_{occtree} \cup T_{pslcore}$. □

The interpretation of situation calculus into the PSL Ontology requires an additional assumption that the set of fluents in any model be finite and bounded.

Theorem 6 *Let FCA be a sentence of the form*

$$(\forall f, s)\ holds(f, s) \supset (f = F_1) \vee ... \vee (f = F_m)$$

There exists an interpretation of $T_{sitcalc} \cup T_{sittime} \cup T_{sitfluent} \cup ACA \cup FCA$ into $T_{disc\_state} \cup T_{occtree} \cup T_{pslcore} \cup WFAS$.

Proof: (Sketch) Suppose

$$\pi_{\mathbf{holds}}(f, s) = holds(f, s)$$

$$\pi_{\mathbf{holds}}(f, S_0) = (\exists s)\ initial(s) \wedge prior(f, s)$$

As with Theorem 4, the techniques introduced in (Doets 1989) and (Backofen, Rogers, & Vijay-Shanker 1995) can be used to show that the models of $T_{disc\_state} \cup T_{occtree} \cup T_{pslcore} \cup WFAS$ are elementarily equivalent to models of $T_{sitcalc} \cup T_{sittime} \cup T_{sitflent} \cup ACA \cup FCA$. □

Although $T_{sitcalc} \cup T_{sittime} \cup T_{sitfluent}$ cannot be interpreted into $T_{disc\_state} \cup T_{occtree} \cup T_{pslcore}$ without the axiom schema, we can show that the two theories are equivalent with respect to a restricted class of first-order sentences.

Theorem 7 *Let $Q(s)$ be a simple state formula in the language of $T_{sitcalc}$ and let $Q'(s)$ be the the image of the formula under the interpretation into $T_{disc\_state} \cup T_{occtree} \cup T_{pslcore}$.*

For any model $\mathcal{M}$ of $T_{sitcalc} \cup T_{sittime} \cup T_{sitfluent}$ there exists a model $\mathcal{N}$ of $T_{disc\_state} \cup T_{occtree} \cup T_{pslcore}$ such that

$$Th(\mathcal{M}) \models (\forall s)\ Q(s) \Leftrightarrow Th(\mathcal{N}) \models (\forall s)\ Q'(s)$$

and

$$Th(\mathcal{M}) \models (\exists s)\ Q(s) \Leftrightarrow Th(\mathcal{N}) \models (\exists s)\ Q'(s)$$

Proof: (Sketch) Axioms 6 and 7 of T_{disc_state} are logically equivalent to the instantiation of the axiom schema $WFAS$ for positive and negative $holds$ literals, respectively. Since simple state formulae are finite boolean combinations of positive and negative $holds$ literals with the same activity occurrence variable, the instantiation of $WFAS$ for a simple state formula is logically equivalent to a finite boolean combination of sentences that are entailed by $T_{disc\_state} \cup T_{occtree} \cup T_{pslcore}$. □

The first sentence in Theorem 7 corresponds to the classical planning problem, while the second sentence corresponds to the entailment of state constraints. By this theorem, the PSL Ontology entails the same set of plans and state constraints as $T_{sitcalc}$.

5 Nondefinability Theorems

In this section, we show that the remaining core theories in the PSL Ontology cannot be interpreted in $T_{sitcalc} \cup T_{sittime}$.

5.1 Automorphisms of Situation Trees

All of the nondefinability theorems rest on the characterization of the automorphisms of situation trees and the failure of these automorphisms to preserve the sets that correspond to the extensions of the functions and relations in models of the PSL Ontology. We introduce three lemmas that characterize properties of the automorphisms of situation trees which will be used in later proofs.

Lemma 3 *Let $\mathcal{R}$ be a model of $T_{sitcalc} \cup T_{sittime}$ and let $Aut(\mathcal{R})$ be the set of automorphisms of $\mathcal{R}$.*

For any $\varphi \in Aut(\mathcal{R})$ and any element $\mathbf{o}$ of the situation tree, $\mathbf{o}$ and $\varphi(\mathbf{o})$ must be on different branches of the situation tree.

Lemma 4 *Let $\mathcal{R}$ be a model of $T_{sitcalc} \cup T_{sittime}$.*

The set of automorphisms $Aut(\mathcal{R})$ of a situation tree is transitive on the set of situations that are the successors of a situation in the tree.

Lemma 5 *Let $\mathcal{R}$ be a model of $T_{sitcalc} \cup T_{sittime}$.*

The set of automorphisms $Aut(\mathcal{R})$ of a situation tree is transitive on the set of actions in $\mathcal{R}$.

5.2 Relationship to Subactivities

The theory $T_{subactivity}$ in PSL Ontology uses the $subactivity$ relation to capture the basic intuitions for the composition of activities[6]. This relation is a discrete partial ordering, in which primitive activities are the minimal elements.

Lemma 6 *A model $\mathcal{M}$ of $T_{subactivity} \cup T_{pslcore}$ with nonprimitive activities is not definable in any model of $T_{sitcalc} \cup T_{sittime}$.*

Proof: We will show that the $\mathbf{subactivity}$ relation in $\mathcal{M}$ is not definable in any model of $T_{sitcalc} \cup T_{sittime}$.

Let $\mathcal{R}$ be a model of $T_{sitcalc} \cup T_{sittime}$.

By Lemma 5, the set of automorphisms $Aut(\mathcal{R})$ of a situation tree is transitive on the set of actions in $\mathcal{R}$; thus, there exists $\varphi \in Aut(\mathcal{R})$ and distinct actions $\mathbf{a_1}, \mathbf{a_2}$ such that $\varphi(\mathbf{a_1}) = \mathbf{a_2}$. By the following axiom of $T_{subactivity}$

$$(\forall a_1, a_2) subactivity(a_1, a_2) \wedge subactivity(a_2, a_1) \supset (a_1 = a_2)$$

we have

$$\langle \mathbf{a_1}, \mathbf{a_2} \rangle \in \mathbf{subactivity} \Rightarrow \langle \varphi(\mathbf{a_1}), \varphi(\mathbf{a_2}) \rangle \notin \mathbf{subactivity}$$

By Proposition 2, the $\mathbf{subactivity}$ relation is not definable in $\mathcal{R}$, and hence $\mathcal{M}$ is not definable in $\mathcal{R}$. □

Theorem 8 *There does not exist an interpretation of $T_{subactivity} \cup T_{pslcore}$ into $T_{sitcalc} \cup T_{sittime}$.*

Proof: This follows from Lemma 6 and Lemma 1. □

[6] The axioms of $T_{subactivity}$ in CLIF can be found at `http://www.mel.nist.gov/psl/psl-ontology/part12/subactivity.th.html`

5.3 Relationship to Atomic Activities

The primary motivation behind the core theory T_{atomic} in the PSL Ontology is to capture intuitions about the occurrence of concurrent activities[7]. The core theory T_{atomic} introduces the function *conc* that maps any two atomic activities to the activity that is their concurrent composition. Essentially, an atomic activity corresponds to some set of primitive activities, so that every concurrent activity is equivalent to the composition of a set of primitive activities.

Lemma 7 *A model $\mathcal{M}$ of $T_{atomic} \cup T_{subactivity} \cup T_{pslcore}$ with nonatomic activities is not definable in any model of $T_{sitcalc} \cup T_{sittime}$.*

Proof: We will show that the $\mathbf{conc}$ function and $\mathbf{atomic}$ relation in $\mathcal{M}$ are not definable in any model of $T_{sitcalc} \cup T_{sittime}$.

Let $\mathcal{R}$ be a model of $T_{sitcalc} \cup T_{sittime}$.

By Lemma 5, the set of automorphisms $Aut(\mathcal{R})$ of a situation tree is transitive on the set of actions in $\mathcal{R}$; thus there exists $\varphi \in Aut(\mathcal{R})$ and actions $\mathbf{a_1}, \mathbf{a_2}, \mathbf{a_3}$ such that $\mathbf{a_3} = \mathbf{conc}(\mathbf{a_1}, \mathbf{a_2})$ and

$$\varphi(\mathbf{a_1}) = \mathbf{a_1},\ \ \varphi(\mathbf{a_2}) = \mathbf{a_2},\ \ \varphi(\mathbf{a_3}) = \mathbf{a_3}$$

It is easy to see that

$$\varphi(\mathbf{conc}(\mathbf{a_1}, \mathbf{a_2})) \neq \mathbf{conc}(\varphi(\mathbf{a_1}), \varphi(\mathbf{a_2}))$$

There also exists $\varphi \in Aut(\mathcal{R})$ and distinct actions $\mathbf{a_1}, \mathbf{a_2}$ such that $\varphi(\mathbf{a_1}) = \mathbf{a_2}$ and

$$\langle \mathbf{a_1}, \mathbf{a_2} \rangle \in \mathbf{subactivity}$$

$$\langle \mathbf{a_1} \rangle \in \mathbf{atomic}, \langle \mathbf{a_2} \rangle \notin \mathbf{atomic}$$

By the following axiom of $T_{subactivity}$

$(\forall a_1, a_2) subactivity(a_1, a_2) \wedge subactivity(a_2, a_1) \supset (a_1 = a_2)$

we have

$$\langle \mathbf{a} \rangle \in \mathbf{atomic} \Rightarrow \langle \varphi(\mathbf{a}) \rangle \notin \mathbf{atomic}$$

By Proposition 2, the $\mathbf{conc}$ function and $\mathbf{atomic}$ relation are not definable in $\mathcal{R}$, and hence $\mathcal{M}$ is not definable in $\mathcal{R}$. □

Theorem 9 *There does not exist an interpretation of $T_{atomic} \cup T_{subactivity} \cup T_{occtree} \cup T_{pslcore}$ into $T_{sitcalc} \cup T_{sittime}$.*

Proof: This follows from Lemma 7 and Lemma 1. □

5.4 Relationship to Complex Activities

The core theory $T_{complex}$ characterizes the relationship between the occurrence of a complex activity and occurrences of its subactivities[8]. Occurrences of complex activities correspond to sets of occurrences of subactivities; in particular, these sets are subtrees of the occurrence tree. An activity tree consists of all possible sequences of atomic subactivity occurrences beginning from a root subactivity occurrence. In a sense, activity trees are a microcosm of the occurrence tree, in which we consider all of the ways in which the world unfolds in the context of an occurrence of the complex activity.

Lemma 8 *A model $\mathcal{M}$ of $T_{complex} \cup T_{atomic} \cup T_{subactivity} \cup T_{pslcore}$ with nonatomic activities such that not all activity occurrences are elements of nontrivial activity trees is not definable in any model of $T_{sitcalc} \cup T_{sittime}$.*

Proof: We will show that the $\mathbf{root}$ and $\mathbf{min\_precedes}$ relations in $\mathcal{M}$ are not definable in any model of $T_{sitcalc} \cup T_{sittime}$.

Let $\mathcal{R}$ be a model of $T_{sitcalc} \cup T_{sittime}$.

By Lemma 4, the set of automorphisms $Aut(\mathcal{R})$ of a situation tree is transitive on the set of situations that are the successors of any situation in the tree.

There exists $\varphi_1 \in Aut(\mathcal{R})$ such that for any $\mathbf{s_1}, \mathbf{s_2}$ that are successors of the same element of the situation tree such that $\varphi_1(\mathbf{s_1}) = \mathbf{s_2}$ and such that $\mathbf{s_1}$ is not an element of any nontrivial activity tree and $\mathbf{s_2}$ is an element of a nontrivial activity tree.

If $\mathbf{s_2}$ is a root of an activity tree, then there exists $\varphi_1 \in Aut(\mathcal{R})$ such that

$$\langle \mathbf{s}, \mathbf{a} \rangle \in \mathbf{root} \Rightarrow \langle \varphi_1(\mathbf{s}), \mathbf{a} \rangle \notin \mathbf{root}$$

If $\mathbf{s_2}$ is not a root of an activity tree, then there exists $\varphi_2 \in Aut(\mathcal{R})$ such that

$$\langle \mathbf{s_1}, \mathbf{s_2}, \mathbf{a} \rangle \in \mathbf{min\_precedes} \Rightarrow$$

$$\langle \varphi_2(\mathbf{s_1}), \varphi_2(\mathbf{s_2}), \varphi_2(\mathbf{a}) \rangle \notin \mathbf{min\_precedes}$$

By Proposition 2, the $\mathbf{root}$ and $\mathbf{min\_precedes}$ relations are not definable in $\mathcal{R}$, and hence $\mathcal{M}$ is not definable in $\mathcal{R}$. □

Theorem 10 *There does not exist an interpretation of $T_{complex} \cup T_{atomic} \cup T_{subactivity} \cup T_{occtree} \cup T_{pslcore}$ into $T_{sitcalc} \cup T_{sittime}$.*

Proof: This follows from Lemma 8 and Lemma 1. □

5.5 Relationship to Complex Activity Occurrences

Within $T_{complex}$, complex activity occurrences correspond to activity trees, and consequently occurrences of complex activities are not elements of the legal occurrence tree. The axioms of the core theory T_{actocc} ensure complex activity occurrences correspond to branches of activity trees[9]. Each complex activity occurrence has a unique atomic root occurrence and each finite complex activity occurrence has a unique atomic leaf occurrence. A subactivity occurrence corresponds to a sub-branch of the branch corresponding to the complex activity occurrence.

[7] The axioms of T_{atomic} in CLIF can be found at `http://www.mel.nist.gov/psl/psl-ontology/part12/atomic.th.html`

[8] The axioms of $T_{complex}$ in CLIF can be found at `http://www.mel.nist.gov/psl/psl-ontology/part12/complex.th.html`

[9] The axioms of T_{actocc} in CLIF can be found at `http://www.mel.nist.gov/psl/psl-ontology/part12/actocc.th.html`

Lemma 9 *A model $\mathcal{M}$ of $T_{actocc} \cup T_{complex} \cup T_{atomic} \cup T_{subactivity} \cup T_{pslcore}$ with occurrences of nonatomic activities is not definable in any model of $T_{sitcalc} \cup T_{sittime}$.*

Proof: We will show that the $\mathbf{subactivity\_occurrence}$ relation in $\mathcal{M}$ is not definable in any model of $T_{sitcalc} \cup T_{sittime}$.

Let $\mathcal{R}$ be a model of $T_{sitcalc} \cup T_{sittime}$.

By Lemma 4, the set of automorphisms $Aut(\mathcal{R})$ of a situation tree is transitive on the set of situations that are the successors of any situation in the tree. Furthermore, $Aut(\mathcal{R})$ only acts on elements of the situation tree, so that it fixes occurrences of complex activities.

By Lemma 3, any $\varphi \in Aut(\mathcal{R})$ maps elements of a branch of the situation tree to another branch of the situation tree; however, the axioms of T_{actoc} entail that all subactivity occurrences of a complex activity occurrences must be on the same branch of the tree. Thus, for any activity occurrence $\mathbf{o_1}$ that is an element of the situation tree and any complex activity occurrence $\mathbf{o_2}$, there exists $\varphi \in Aut(\mathcal{R})$ such that

$$\langle \mathbf{o_1}, \mathbf{o_2} \rangle \in \mathbf{subactivity\_occurrence} \Rightarrow$$

$$\langle \varphi(\mathbf{o_1}), \mathbf{o_2} \rangle \notin \mathbf{subactivity\_occurrence}$$

By Proposition 2, the $\mathbf{subactivity\_occurrence}$ relation is not definable in $\mathcal{R}$, and hence $\mathcal{M}$ is not definable in $\mathcal{R}$. □

Theorem 11 *There does not exist an interpretation of $T_{actocc} \cup T_{complex} \cup T_{atomic} \cup T_{subactivity} \cup T_{occtree} \cup T_{pslcore}$ into $T_{sitcalc} \cup T_{sittime}$.*

Proof: This follows from Lemma 9 and Lemma 1. □

6 Summary

In this paper we have characterized the relationship between the PSL Ontology and two other ontologies – a time ontology and Reiter's second-order axiomatization of situation calculus. With the addition of a first-order axiom schema and the restriction to finite domains of activities and fluents, elements of the PSL Ontology are elementarily equivalent to models of the situation calculus axiomatization. Furthermore, the core theories in PSL Ontology that axiomatize subactivities and complex activities are not definable in the situation calculus.

References

Backofen, R.; Rogers, J.; and Vijay-Shanker, K. 1995. A first-order axiomatization of the theory of finite trees. *Journal of Logic, Language, and Information* 4:5–39.

Bock, C., and Gruninger, M. 2005. PSL: A semantic domain for flow models. *Software and Systems Modeling* 4:209–231.

Doets, K. 1989. Monadic π^1_1-theories of π^1_1-properties. *Notre Dame Journal of Formal Logic* 30:224–240.

Enderton, H. 1972. *Mathematical Introduction to Logic*. Academic Press.

Gruninger, M. 2004. Ontology of the Process Specification Language. In Staab, S., and Studer, R., eds., *Handbook of Ontologies in Information Systems*. Springer-Verlag.

Hayes, P. 1996. A catalog of temporal theories. Technical Report UIUC-BI-AI-96-01, University of Illinois.

Levesque; H., Reiter, R.; Lesperance, Y.; Lin, F.; and Scherl, R. 1997. Golog: A logic programming language for dynamic domains. *Journal of Logic Programming* 31:92–128.

Marker, D. 2002. *Model Theory: An Introduction*. Springer-Verlag.

Pinto, J., and Reiter, R. 1995. Reasoning about time in the situation calculus. *Annals of Mathematics and Artificial Intelligence* 14:2510–268.

Reiter, R. 2001. *Knowlege in action: logical foundations for specifying and implementing dynamical systems*. Cambridge, MA, USA: MIT Press.

Ontologies and Domain Theories

Michael Grüninger
Department of Mechanical and Industrial Engineering
University of Toronto
gruninger@mie.utoronto.ca

Abstract

Although there is consensus that a formal ontology consists of a set of axioms within some logical language, there is little consensus on how a formal ontology differs from an arbitrary theory. There is an intuitive distinction between the formal ontology and the set of domain theories that use the ontology, but there has been no characterization of this distinction in the context of first-order ontologies. In this paper we utilize the notions of definable sets and types from model theory mathematical logic to provide a semantic characterization of the domain theories for an ontology. We illustrate this approach with respect to several formal ontologies from mathematical logic and knowledge representation.

1 Motivation

Ontological engineering was born with the promise of reusability, integration, and interoperability. Of increasing importance are the problems merging ontologies from different domains and translating among multiple ontologies from the same domain. An obstacle to achieving this vision has been a lack of consensus over the nature of the axioms within a formal ontology. On the one hand, formal ontologies are specific theories – we are not defining new languages or logics. On the other hand, formal ontologies are different from arbitrary theories in that we intuitively think of ontologies as being the reusable portion of domain theories. This begs the question of defining domain theories, and it raises the perennial debate of the difference between ontologies and knowledge bases.

In the course of providing a formal characterization of domain theories for ontologies, we are guided by several intuitions.

- Domain theories and queries are constructed using ontologies – typical reasoning problems include sentences that describe a particular scenario in addition to the axioms of the ontologies.
- Ontologies are the reusable parts of domain theories, in the sense that all domain theories for an ontology are extensions of a unique set of axioms in the ontology.
- In semantic interoperability scenarios, software applications exchange sentences that are written using ontologies, rather than exchange axioms from the ontologies themselves.

The objective of this paper to is to provide a semantic characterization of domain theories, that is, one that is based on properties of the models of the formal ontology.

1.1 Some Motivating Examples

We consider several ontologies and the sentences that are intuitively their domain theories. We begin with two mathematical theories which are well understood before moving on to two ontologies from the knowledge representation community.

Algebraically Closed Fields Suppose that two software applications share the ontology of algebraically closed fields (Hodges 1993), for example, CAD software that is based on algebraic geometry. Such software applications will exchange shapes that are specified by polynomials; they are not exchanging axioms in the ontology. In this case, we can see that the domain theories for algebraically closed fields are polynomials.

Groups Domain theories for the ontology of groups (Hodges 1993) are either explicitly specifying particular groups or subgroups of other groups. A group presentation defines a group by specifying a set of elements of a group (known as generators) such that all other elements of the group can be expressed as the product of the generators subject to a set of equations (known as relations among the generators). For example, the group presentation for the cyclic group of order three is the equation $\mathbf{a} \cdot \mathbf{a} \cdot \mathbf{a} = 1$, and it is equivalent to the theory of the group with respect to the element $\mathbf{a}$ in the domain.

Time Ontologies Consider an ontology of time T_{dense} (Hayes 1996) in which the set of timepoints is linearly ordered and dense. Such an ontology is typically used to specify the underlying constraints in commonsense reasoning problems about events (e.g. "Bob left home before arriving at work and Alice arrived at work after Bob"). This set of constraints constitutes a domain theory for the ontology T_{dense}; in general, the domain theories consist of boolean combinations of sets of timepoints that form intervals on the linear ordering.

Situation Calculus The axiomatization of situation calculus in (Reiter 2001) includes a set of foundational axioms (the ontology $T_{sitcalc}$) together with a set of axioms which plays the role of a domain theory.

A simple state formula is a formula which contains a unique situation variable and which contains only $holds$ literals. A precondition axiom for an activity A is a sentence of the form

$$(\forall s)\, poss(A, s) \supset Q(s)$$

where $Q(s)$ is a simple state formula. An effect axiom for an activity A is a sentence of the form

$$(\forall s)\, Q_1(s) \supset holds(F, do(A, s))$$

where $Q(s)$ is a simple state formula and F is a fluent. Basic action theories, which consist of sets of precondition and effect axioms, are domain theories for situation calculus.

2 Domain Theories and Definable Sets

The characterization of ontologies and domain theories rests on the model-theoretic notion of definability. After introducing this notion, we will use it to distinguish between the different classes of theories within an ontology.

2.1 Definable Sets

We will adopt the following definition from (Marker 2002):

Definition 1 *Let $\mathcal{M}$ be a structure in a language L.*

A set $X \subseteq M^n$ is definable in $\mathcal{M}$ iff there is a formula $\varphi(v_1, ..., v_n, w_1, ..., w_m)$ of L and $\overline{\mathbf{b}} \in M^m$ such that

$$X = \{\overline{\mathbf{a}} \in M^n \ : \ \mathcal{M} \models \varphi(\overline{a}, \overline{b})\}$$

X is A-definable if there is a formula $\varphi(\overline{v}, w_1, ..., w_l)$ and $\overline{\mathbf{b}} \in A^l$ such that

$$X = \{\overline{\mathbf{a}} \in M^n \ : \ \mathcal{M} \models \varphi(\overline{\mathbf{a}}, \overline{\mathbf{b}})\}$$

We say that X is $\emptyset$-definable if $A = \emptyset$. If A is nonempty, we say that X is definable with parameters.

Example 1 *Suppose $\mathcal{M}$ is a discretely ordered ring.*

The set of even numbers is $\emptyset$-definable in $\mathcal{M}$:

$$\{x \ : \ (\exists y)\, x = y + y\}$$

The set of prime numbers is $\emptyset$-definable in $\mathcal{M}$:

$$\{x \ : \ (\forall y, z)\, (y \cdot z = x) \supset (y = x) \vee (z = x)\}$$

The set

$$\{x \ : \ a_0 + a_1 x + a_2 x^2 + ... + a_n x^n = 0\}$$

is definable with parameters $a_0, ..., a_n$.

2.2 Definitional Extensions and Core Theories

An ontology is specified by a set of axioms in some formal language. Nevertheless, this is not an amorphous set, and the notion of definability allows us to distinguish between different kinds of sentences within an ontology.

Definition 2 *A theory T_1 is a definitional extension of a theory T iff every constant, function, and relation in models of T_1 is $\emptyset$-definable in models of T.*

It is easy to see that a definitional extension of a theory T is also a conservative extension of T, although the converse is not true; that is, there are conservative extensions of theories which are not definitional extensions.

Definition 3 *A theory T_{core} is a core theory iff no constant, function, or relation in models of T_{core} is definable in the models of any other theory.*

Combining these two classes of sentences gives us the following definition of an ontology.

Definition 4 *An ontology T_{onto} is a theory consisting of a set of core theories and a set of definitional extensions.*

Intuitively, the core theories axiomatize the primitive functions and relations in the ontology. If a core theory in an ontology is an extension of some other core theories in the ontology, then it is a nonconservative extension.

In the case of the PSL Ontology ((Gruninger 2004), (Gruninger & Kopena 2004)), the definitional extensions within the ontology are axiomatizations of the classes of activities and activity occurrences that correspond to values of the invariants that are used to classify the models of the core theories within the ontology.

If we consider the examples from Section 1.1, we can see that an ontology is not an arbitrary set of sentences. In the case of algebraically closed fields, polynomials are sentences that are not in a core theory or definitional extension. Similarly, precondition and effect axioms are not part of a core theory or definitional extension. We therefore require a precise definition of the class of sentences that correspond to domain theories.

2.3 Domain Theories

We are still faced with the question of how domain theories are different from the other two classes of theories within an ontology. Whereas a definitional extension is an axiomatization of the $\emptyset$-definable sets in a model of an ontology T_{onto}, we will say that a domain theory for an ontology T_{onto} is an axiomatization of sets that are definable with parameters in some model of T_{onto}.

Definition 5 *A theory T_{dt} is a domain theory for an ontology T_{onto} iff every formula in T_{dt} defines a set $X \subseteq M^n$ with parameters in some model $\mathcal{M}$ of T_{onto}.*

In general, domain theories are not conservative extensions of the ontology. For example, the domain theory consisting of the equations

$$(\mathbf{a} \cdot (\mathbf{a} \cdot \mathbf{a})) = 1$$

in the theory of groups entails the sentence

$$(\exists x, y)\, ((x \cdot y) = (y \cdot x)) \wedge (x \neq 1) \wedge (y \neq 1)$$

which is not entailed by the axioms in the theory of groups alone.

On the other hand, domain theories are distinct from arbitrary nonconservative extensions of the ontology. For example, the sentence

$$(\forall x, y)\, (x \cdot y) = (y \cdot x)$$

axiomatizes abelian groups; it forms a nonconservative extension of the theory of groups, yet we would not consider it to be a domain theory, since it does not define any sets with parameters in some model of group theory.

This approach to characterizing the sentences in an ontology generalizes a distinction long made within the description logic community – sentences in the ABox are domain theories, subsumption axioms in the TBox are contained in core theories, and equivalence axioms are part of the definitional extensions of the ontology.

3 Domain Theories and Types

The next step is to show how the set of domain theories for an ontology can be characterized with respect to properties of the models of the ontology. This will allow us to formalize the intuitions presented earlier in Section 1.

3.1 Types

Types describe a model of a theory from the point of view of a single element or a finite set of elements ((Marker 2002), (Rothmaler 2000)).

Definition 6 *Let $\mathcal{M}$ be a model for a language $\mathcal{L}$.*

The type of an element $\mathbf{a} \in M$ is defined as $type_{\mathcal{M}}(\mathbf{a}) := \{\phi : \phi$ is a formula of $\mathcal{L}, \mathcal{M} \models \phi(a)\ \}$

An n-type for a theory T is a set $\Phi(x_1, ..., x_n)$ of formulae, such that for some model $\mathcal{M}$ of T, and some n-tuple $\overline{a}$ of elements of $\mathcal{M}$, we have $\mathcal{M} \models \phi(\overline{a})$ for all ϕ in Φ.

If t is an n-type, then a model $\mathcal{M}$ realizes t iff there are $a_1, ..., a_n \in M$ such that

$$\mathcal{M} \models t(a_1, ..., a_n)$$

A type p is a complete n-type iff $\phi \in p$ or $\neg\phi \in p$ for any formula ϕ with n free variables; a partial type is a type that is not complete.

Informally, the type for an element in a model is a set of formulae which are satisfied by the element in the model. An n-type for a theory is a consistent set of formulae (each of which has n free variables) which is satisfied by a model of the theory.

3.2 Characterization Theorems for Domain Theories

The model-theoretic notion of type allows us to formalize the intuition that domain theories are theories about elements in the domain of a model of the ontology.

Theorem 1 *A set of sentences T_{dt} is a domain theory for an ontology T_{onto} iff it is logically equivalent to a boolean combination of finite partial n-types for T_{onto}.*

Proof: ⇒:) Let $\varphi(x_1, ..., x_n)$ be a sentence in a domain theory for T_{onto} and let

$$\{\overline{a} \ : \ \mathcal{M} \models \varphi(\overline{a})\}$$

be the set defined by this sentence in a model $\mathcal{M}$ of T_{onto}. It is easy to see that this set realizes the finite n-type $\varphi(x_1, ..., x_n)$ in $\mathcal{M}$.

⇐:) The set of elements that realize a finite type in $\mathcal{M}$ constitute a definable set. The boolean combination of finite partial n-types is equivalent to the union, intersection, complement, and projection of definable sets, and these operations preserve definable sets. Therefore, the boolean combination of n-types is logically equivalent to a domain theory. □

This result shows that we can specify all possible domain theories for an ontology by identifying the finite partial types for elements in the models of the ontology.

Not all types correspond to domain theories, since a type that consists of an infinite set of formulae may not be first-order definable. For example,

$$\{0 < c, S(0) < c, S(S(0)) < c, ...\}$$

is an infinite type that is realized by a nonstandard number $\mathbf{c}$ in a model of $Th(\mathbb{N}, 0, S, <)$, yet the set is not first-order definable in the theory.

The next two theorems characterize domain theories with respect to the models of the ontology, and formalize the intuition that ontologies are the reusable parts of domain theories, in the sense that all domain theories for an ontology are extensions of a unique set of axioms in the ontology.

Theorem 2 *If T_{dt} is a domain theory for an ontology T_{onto} then there exists a model $\mathcal{M}$ of T_{onto} such that*

$$T_{onto} \cup T_{dt} \subseteq Th(\mathcal{M})$$

Proof: By Definition 5, the sentences in T_{dt} define sets with parameters in some model $\mathcal{M}$ of T_{onto}. We therefore have

$$T_{onto} \subseteq Th(\mathcal{M})$$

Suppose that there is a sentence $\Sigma \in T_{dt}$ such that $\Sigma \notin Th(\mathcal{M})$. In this case, we must have $\mathcal{M} \models \neg\Sigma$, which would mean that Σ does not define a set in $\mathcal{M}$, and hence would not be a sentence in a domain theory. We therefore also have

$$T_{dt} \subseteq Th(\mathcal{M})$$

□

From this result we can see that models of a domain theory are models of the ontology.

Theorem 3 *For any model $\mathcal{M}$ of T_{onto}, there exists a domain theory T_{dt} for T_{onto} such that*

$$T_{onto} \cup T_{dt} \subseteq Th(\mathcal{M})$$

Proof: Since $\mathcal{M}$ is a model of T_{onto}, we have

$$T_{onto} \subseteq Th(\mathcal{M})$$

If T_{dt} is the set of sentences that define sets in $\mathcal{M}$, then $T_{dt} \neq \emptyset$ (since any finite set is definable). T_{dt} is therefore a domain theory such that

$$\mathcal{M} \models T_{dt}$$

As a result, we know that $T_{onto} \cup T_{dt}$ is consistent.

Since $\mathcal{M} \models T_{onto} \cup T_{dt}$, we have

$$T_{onto} \cup T_{dt} \subseteq Th(\mathcal{M})$$

□

Note that any definable set must have some axiomatization, whereas nondefinable sets cannot be axiomatized by any theory. Furthermore, every model contains definable sets (since finite sets are always definable). Consequently, domain theories will always exist for any ontology.

We can define a complete domain theory as one that satisfies

$$T_{onto} \cup T_{dt} = Th(\mathcal{M})$$

for some model $\mathcal{M}$ of T_{onto}. In other words, a complete domain theory is an axiomatization of a particular model of the ontology. Not all ontologies will have complete domain theories. For example, there exist infinite groups that do not have a finite presentation. The standard models of more powerful ontologies, such as Peano Arithmetic and the theory of the free semigroups, are not axiomatizable, so that any domain theory in such cases would be incomplete.

3.3 Techniques for Specifying Domain Theories

Model theory provides several techniques for specifying the types for first-order theories. The most widely use technique is known as the elimination of quantifiers, in which one focusses on the sets that are definable by formulae that are quantifier-free.

A theory T admits the elimination of quantifiers if for every formula ϕ there is a formula ψ such that

$$T \models \phi \equiv \psi$$

One typically determines this by specifying a set of quantifier-free formulae Δ (known as the elimination set) such that for every formula ϕ in the language of T there is a formula ψ which is a boolean combination of formulae in Δ, and ϕ is equivalent to ψ in every model of T. It is easy to see that in ontologies that admit elimination of quantifiers, the elimination set characterizes the set of types.

Unfortunately, not all ontologies admit the elimination of quantifiers, and the characterization of the definable sets and types realized in models of these ontologies can become quite complicated.

3.4 Revisiting the Examples

The set of types for many ontologies within mathematical logic have been specified within the literature. We can see that the types for the ontologies that we considered in Section 1 do indeed correspond to the intuitions that we have about their domain theories.

Algebraically Closed Fields and Polynomials Since algebraically closed fields admit the elimination of quantifiers, it can be shown ((Marcja & Toffalori 2003)) that any irreducible polynomial corresponds to a complete 1-type and that 2-types correspond to algebraic curves. In other words, there is a one-to-one correspondence between the set of roots of polynomials (algebraic numbers) and definable elements in the models of T_{field}. There is also the complete 1-type that is realized by all numbers that are transcendental over models of the ontology; this type is not generated by a finite set of formulae ,and hence does not correspond to a domain theory.

Presentations and Groups Although the theory of groups does not admit elimination of quantifiers, it can be shown that all 1-types for T_{group} are of the form

$$(\exists y, z)\, x = y \cdot z$$

We can see that both presentations and group equations are domain theories for groups, since they are boolean combinations of 1-types. In a sense, the presentation is equivalent to the types realized by all elements of the group G; when a presentation exists, it is a complete axiomatization of the theory $Th(G)$ for the group.

Time Ontologies Models of T_{dense} are isomorphic to dense linear orderings, whose n-types have been fully characterized in (Rosenstein 1982). The n-types for T_{dense} are therefore boolean combinations of literals of the form $before(v_i, v_j)$ and $v_i = v_j$. Thus the types for dense linear orderings correspond to the domain theories discussed in Section 1.1.

Action Theories in Situation Calculus Although there has been no work on the characterization of the types for $T_{sitcalc}$ we can still show that action theories define sets in models of $T_{sitcalc}$, and so are domain theories for $T_{sitcalc}$.

The precondition axiom for each action $\mathbf{a}$ is realized by the definable set of situations

$$\{\mathbf{s_1} \,:\, \mathbf{s_1} = \mathbf{do}(\mathbf{a}, \mathbf{s}), \langle \mathbf{s_1} \rangle \in \mathbf{executable}\}$$

that is, the set of executable situations that correspond to occurrences of $\mathbf{a}$. The effect axiom for each action $\mathbf{a}$ is realized by the definable set of situations

$$\{\mathbf{s_1} \,:\, \mathbf{s_1} = \mathbf{do}(\mathbf{a}, \mathbf{s}), \langle \mathbf{f}, \mathbf{s_1} \rangle \in \mathbf{holds} \Leftrightarrow \langle \mathbf{f}, \mathbf{s} \rangle \notin \mathbf{holds}\}$$

that is, the set of situations that achieve or falsify specific fluents. A complete characterization of all types and domain theories for $T_{sitcalc}$ is an open research problem.

4 Evaluating the Ontology

We can evaluate the correctness and completeness of the ontology and domain theories with respect to the characterization of definable sets. For correctness, all domain theories for an ontology must be consistent with the ontology. For completeness, we need to determine whether or not there exist models of the ontology that do not realize any types corresponding to some class of domain theories.

Definition 7 *Let Σ be a set of types for a theory T.*

T is definably complete with respect to Σ iff every model of T realizes some type in Σ.

In $T_{sitcalc}$, precondition axioms are domain theories, but not all activities realize precondition axioms i.e. there are other classes of domain theories

Theorem 4 *The ontology $T_{sitcalc}$ is not definably complete with respect to the set of basic action theories.*

Proof: We can construct a model of $T_{sitcalc}$ that does not satisfy any basic action theory (i.e. set of precondition and effect axioms).

Let $\mathbf{s_1}, \mathbf{s_2}$ be situations in the situation tree that agree on state, that is, for any fluent $\mathbf{f}$,

$$\langle \mathbf{f}, \mathbf{s_1} \rangle \in \mathbf{holds} \Leftrightarrow \langle \mathbf{f}, \mathbf{s_2} \rangle \in \mathbf{holds}$$

Now specify the extension of the $\mathbf{poss}$ relation for an activity $\mathbf{a}$ such that

$$\langle \mathbf{a}, \mathbf{s_1} \rangle \in \mathbf{poss}, \langle \mathbf{a}, \mathbf{s_2} \rangle \notin \mathbf{poss}$$

The activity $\mathbf{a}$ cannot realize any precondition axiom, since the same simple state formula is realized by both $\mathbf{s_1}$ and $\mathbf{s_2}$.

Now specify the extension of the $\mathbf{holds}$ relation for the activity $\mathbf{a}$ such that

$$\langle \mathbf{f}, \mathbf{do}(\mathbf{a}, \mathbf{s_1}) \rangle \in \mathbf{holds}, \langle \mathbf{f}, \mathbf{do}(\mathbf{a}, \mathbf{s_2}) \rangle \notin \mathbf{holds}$$

The activity $\mathbf{a}$ cannot realize any effect axiom, since the same simple state formula is realized by both $\mathbf{s_1}$ and $\mathbf{s_2}$. □

On the other hand, the PSL Ontology explicitly axiomatizes the classes of activities that realize the types corresponding to basic action theories[1]

Theorem 5 *Let MAA (Markovian Activity Assumption) be the sentence*

$$(\forall a) activity(a) \supset markov\_precond(a) \wedge markov\_effect(a)$$

The ontology $T_{disc\_state} \cup T_{occtree} \cup T_{pslcore} \cup MAA$ is definably complete with respect to the set of basic action theories.

It should be noted that $T_{disc\_state} \cup T_{occtree} \cup T_{pslcore}$ alone is not definably complete, since there are models that do not realize precondition and effect axioms; on the other hand, all models of $T_{disc\_state} \cup T_{occtree} \cup T_{pslcore} \cup MAA$ realize precondition and effect axioms.

It must be emphasized that one cannot specify domain theories using axiom schemata – there will typically be mutually inconsistent domain theories for the same ontology, yet the union of sentences that are instantiations of an axiom schema must be consistent. For example, both of the following sentences satisfy the syntactic definition of precondition axioms in situation calculus

$$(\forall s)\, poss(A, s) \supset holds(F, s)$$

$$(\forall s)\, poss(A, s) \supset \neg holds(F, s)$$

yet they are mutually inconsistent.

We can also use this approach to show that some approaches to process ontologies are in fact specifying classes of domain theories rather than ontologies. For example, the axiomatization of actions and events in (Allen & Ferguson 1994) does not include any core theories or definitional extensions; it only contains a specification of the classes of sentences that constitute event definitions, action definitions, and event generation axioms.

5 Classifying Domain Theories

We can use the notion of definable completeness of an ontology to classify the domain theories for the ontology. In particular, we can classify domain theories with respect to the sets that are $\emptyset$-definable by the sentence Φ such that $T_{onto} \cup \Phi$ is definably complete with respect to the domain theories.

For example, by Theorem 5, $T_{disc\_state} \cup T_{occtree} \cup T_{pslcore} \cup MAA$ is definably complete; activities in the set defined by the sentence MAA realize the types corresponding to basic action theories. Activities that are not in the set (that is, activities that do not satisfy the sentence MAA) do not realize the types corresponding to basic action theories. This gives a model-theoretic definition of basic action theories, rather than simply a syntactic definition.

Within the PSL Ontology, sentences such as MAA axiomatize invariants that are used to classify the models of the core theories (Gruninger & Kopena 2004). Invariants are properties of models that are preserved by isomorphism. For some classes of structures, invariants can be used to classify the structures up to isomorphism; for example, vector spaces can be classified up to isomorphism by their dimension. For other classes of structures, such as graphs, it is not possible to formulate a complete set of invariants. Nevertheless, even without a complete set, invariants can still be used to provide a classification of the models of a theory.

In general, the set of models for the core theories of an ontology are partitioned into equivalence classes defined with respect to the set of invariants of the models. Each equivalence class in the classification of the models of the ontology is axiomatized using a definitional extension of the ontology. Each definitional extension in the ontology is associated with a unique invariant; the different classes of objects that are defined in an extension correspond to different properties of the invariant. In this way, the terminology of the ontology arises from the classification of the models of the core theories with respect to sets of invariants.

Using this approach, the classification of domain theories mirrors the classification of the models of the core theories, as well as the organization of the definitional extensions within the ontology.

6 Reasoning Problems

Many reasoning problems with ontologies (such as decision problems for mathematical theories) incorporate domain theories as well as the set of axioms in the ontologies themselves.

The Word Problem in group theory is specified for a particular group and it requires both the axioms for groups as well as the presentation for the group:

$$T_{group} \cup \Sigma_{presentation} \models (w = 1)$$

The query in this case determines whether the product of group elements w is equal to the identity element in the group.

In a temporal reasoning problem, we consider a particular scenario of temporal constraints in addition to the axioms for

[1] The axiomatization of $markov\_precond$ in CLIF (Common Logic Interchange Format) can be found at `http://www.mel.nist.gov/psl/psl-ontology/part42/state_precond.def.html`
The axiomatization of $markov\_effect$ in CLIF can be found at `http://www.mel.nist.gov/psl/psl-ontology/part42/state_effects.def.html`

the time ontology, and determine whether or not a particular temporal constraint is entailed by the scenario:

$$T_{time} \cup \Sigma_{scenario} \models before(T_1, T_2)$$

For situation calculus, the antecedent of a reasoning problem such as planning includes basic action theories, while the query sentence is an existentially quantified simple state formula:

$$T_{sitcalc} \cup \Sigma_{action} \models (\exists s)\; Q(s)$$

In general, an entailment problem for an ontology T_{onto} has the form

$$T_{onto} \cup \Sigma_{dt} \models \Sigma_{query}$$

where Σ_{dt} is a domain theory for T_{onto} and Σ_{query} is a sentence in the language of the ontology. This leads to the next question – what class of sentences in the language of the ontology characterize the query?

Any sentence in such a query (that is, a sentence in Σ_{query}) can also be considered to be a domain theory. For example, in the word problem for groups, the query sentence is a group equation, which is a type for the theory of groups. Similarly, simple state formulae are types for fluents in situation calculus.

We can provide a model-theoretic characterization of queries using the following notion:

Definition 8 *A type p is isolated iff there is a formula $\varphi \in p$ such that for any $\psi \in p$, we have*

$$T \models (\forall \overline{v})\; \varphi(\overline{v}) \supset \psi(\overline{v})$$

Queries therefore correspond to nonisolated types for the ontology. Using this definition, we can also consider queries to be weak domain theories, in the sense that they are entailed by other domain theories. We can therefore apply the earlier techniques for arbitrary domain theories to provide a characterization of the possible queries in reasoning problems that use a particular ontology.

The same techniques that were used to characterize all possible domain theories for an ontology by specifying the types for the ontology can be used to characterize the queries by specifying the nonisolated types for the ontology. We can also classify the queries for an ontology by characterizing the additional sentences that are required in order for an ontology to be definably complete with respect to the class of queries.

7 Summary

Although there is an intuitive distinction between the formal ontology and the set of domain theories that use the ontology, there has been no characterization of this distinction. In this paper we have utilized the notions of definable sets and types from model theory mathematical logic to provide a semantic characterization of the domain theories for an ontology that gives a clear logical distinction between ontologies and domain theories.

Domain theories for an ontology are the axiomatization of definable sets in models of the ontology. This is equivalent to saying that a domain theory for an ontology is a boolean combination of finite partial n-types for the ontology.

The model-theoretic characterization of domain theories serves as an evaluation criterion for ontologies, which can in turn be used to classify the domain theories for an ontology.

This approach lays the groundwork for a comprehensive methodology for the evaluation of formal ontologies by specifying the complete sets of n-types that are realized in models of the ontologies.

References

Allen, J. F., and Ferguson, G. 1994. Actions and events in interval temporal logic. *Journal of Logic and Computation* 4:531–579.

Gruninger, M., and Kopena, J. 2004. Semantic integration through invariants. *AI Magazine* 26:11–20.

Gruninger, M. 2004. Ontology of the Process Specification Language. In Staab, S., and Studer, R., eds., *Handbook of Ontologies in Information Systems*. Springer-Verlag.

Hayes, P. 1996. A catalog of temporal theories. Technical Report UIUC-BI-AI-96-01, University of Illinois.

Hodges, W. 1993. *Model Theory*. Cambridge University Press.

Marcja, A., and Toffalori, C. 2003. *A Guide to Classical and Modern Model Theory*. Kluwer Academic Publishers.

Marker, D. 2002. *Model Theory: An Introduction*. Springer-Verlag.

Reiter, R. 2001. *Knowlege in action: logical foundations for specifying and implementing dynamical systems*. Cambridge, MA, USA: MIT Press.

Rosenstein, J. 1982. *Linear Orderings*. Academic Press.

Rothmaler, P. 2000. *Introduction to Model Theory*. Taylor and Francis.

Order-Sorted Reasoning in the Situation Calculus

Yilan Gu
Department of Computer Science
University of Toronto
10 King's College Road
Toronto, ON, M5S 3G4, Canada
Email: yilan@cs.toronto.edu

Mikhail Soutchanski
Department of Computer Science
Ryerson University
245 Church Street, ENG281
Toronto, ON, M5B 2K3, Canada
Email: mes@scs.ryerson.ca

Abstract

We propose a theory for reasoning about actions based on order-sorted predicate logic where one can consider an elaborate taxonomy of objects. We are interested in the projection problem: whether a statement is true after executing a sequence of actions. To solve it we design a regression operator that takes advantage of well-sorted unification between terms. We show that answering projection queries in our logical theories is sound and complete with respect to that of in Reiter's basic action theories. Moreover, we demonstrate that our regression operator based on order-sorted logic can provide significant computational advantages in comparison to Reiter's regression operator.

Introduction

In his influential paper (Hayes 1971) titled "A Logic of Actions", Pat Hayes proposed an outline of a logical theory for reasoning about actions based on many-sorted logic with equality. His paper inspired subsequent work on many-sorted logics in AI. In particular, A. Cohn (Cohn 1987; 1989) developed expressive many-sorted logic and reviewed all previous work in this area. Reasoning about actions based on the situation calculus has been extensively developed in (Reiter 2001). However, he considers a logical language with sorts for actions, situations and just one catch-all sort $Object$ for the rest that remains unelaborated. Surprisingly, even if the idea proposed by Hayes seems straightforward, there is still no formal study of logical and computational properties of a version of the situation calculus with many related sorts for objects in the domain. Perhaps, this is because mathematical proofs of these properties are not straightforward. We undertake this study and demonstrate that reasoning about actions with elaborated sorts has significant computational advantages in comparison to reasoning without them. In contrast to an approach to many-sorted reasoning (Schmidt 1938; Wang 1952; Herbrand 1971) where variables of different sorts range over unrelated universes, we consider a case when sorts are related to each other, so that one can construct an elaborated taxonomy. This is often convenient for representation of commonsense knowledge about a domain.

Generally speaking, we are usually interested in a comprehensive taxonomic structure for sorts, where sorts may inherit from each other and may have non-empty intersections. Hence, we consider formulating the situation calculus in an order-sorted (predicate) logic to describe taxonomic information about objects. We are interested in the projection problem (whether a statement is true after executing a sequence of actions) and we would like to use regression to solve this problem (Reiter 2001). Note that even if both many-sorted logic and order-sorted logic can be translated to unsorted, using order-sorted logic can bring about significant computational advantages, for example in deduction. This was a primary driving force for (Walther 1987) and (Cohn 1987). We show that regression in order-sorted SC can benefit from well-sorted unification. One can gain computational efficiency by terminating regression steps earlier when objects of incommensurable sorts are involved.

It is well-known that *PDDL* supports typed (sorted) variables and many implemented planners can take advantage of types (Ghallab *et al.* 1998). (Classen *et al.* 2007) proposes formal semantics for the typed ADL subset of PDDL using ES, a dialect of SC. Our paper focus on the relations between Reiter's BATs and our new order-sorted BATs and the computational advantages which regression in order-sorted BATs can provide (sometimes). We contribute towards a formal logical foundation of PDDL.

Background

In general, order-sorted logic (OSL) (Oberschelp 1962; 1990; Walther 1987; Schmidt-Schauβ 1989; Bierle *et al.* 1992; Weidenbach 1996) restricts the domain of variables to subsets of the universe (i.e., *sorts*). Notation $x\!:\!Q$ means that variable x is of sort Q and $\mathbf{V}_Q$ is the set of variables of sort Q. For any n, sort cross-product $Q_1 \times \cdots \times Q_n$ is abbreviated as $\vec{Q}_{1..n}$; term vector $t_1, \ldots, t_n$ is abbreviated as $\vec{t}_{1..n}$; variable vector $x_1, \ldots, x_n$ is abbreviated as $\vec{x}_{1..n}$; and, variable declaration sequence $x_1\!:\!Q_1, \ldots, x_n\!:\!Q_n$ is abbreviated as $\vec{x}_{1..n}\!:\!\vec{Q}_{1..n}$.

A theory in OSL always includes a set of declarations (called *sort theory*) to describe the hierarchical relationships among sorts and the restrictions on ranges of the arguments of predicates and functions. In particular, a sort theory $\mathcal{T}$ includes a set of *term declarations* of the form $t : Q$ representing that term t is of sort Q, *subsort declarations* of the form $Q_1 \leq Q_2$ representing that sort Q_1 is a (direct) subsort of sort Q_2 (i.e., every object of sort Q_1 is also of sort Q_2), and *predicate declarations* of the form $P\!:\!\vec{Q}_{1..n}$ representing that the i-th argument of the n-ary predicate P is of sort Q_i for $i = 1..n$. A *function declaration* is a special term declaration where term t is a function with distinct variables as arguments: for each n-ary function f, the abbreviation of its function declaration is of the form $f : Q_{1..n} \rightarrow Q$, where Q_i is the sort of the i-th argument of f and Q is the sort of the value of f. $c\,{:}\,Q$ is a special function declaration, representing that constant c is of sort Q. Arguments of equality "=" can be of any sort. Below, we consider a *finite simple* sort theory only, in which there are finitely many sorts and declarations, the term declarations are all function declarations, and for each function there is one and only one declaration.

For any sort theory $\mathcal{T}$, subsort relation $\leq_{\mathcal{T}}$ is a partial ordering defined by the reflexive and transitive closure of the subsort declarations. Then, following the standard terminology of lattice theory, if each pair of sort symbols in $\mathcal{T}$ has greatest lower bound (g.l.b.), then we say that *the sort hierarchy of $\mathcal{T}$ is a meet semi-lattice* (Walther 1987). Moreover, a *well-sorted term* (wrt $\mathcal{T}$) is either a sorted variable, or a constant declared in $\mathcal{T}$, or a functional term $f(\vec{t}_{1..n})$, in which each t_i is well-sorted and the sort of t_i is a subsort of Q_i, given that $f : \vec{Q}_{1..n} \rightarrow Q$ is in $\mathcal{T}$. A *well-sorted atom* (wrt $\mathcal{T}$) is an atom $P(\vec{t}_{1..n})$ (can be $t_1 = t_2$), where each t_i is a well-sorted term of sort Q'_i, and $Q'_i \leq_{\mathcal{T}} Q_i$, given that $P : \vec{Q}_{1..n}$ is in $\mathcal{T}$. A *well-sorted formula* (wrt $\mathcal{T}$) is a formula in which all terms (including variables) and atoms are well-sorted. Any term or formula that is not well-sorted is called *ill-sorted*. A *well-sorted substitution* (wrt $\mathcal{T}$) is a substitution ρ s.t. for any variable $x : Q$, ρx (the result of applying ρ to x) is a well-sorted term and its sort is a (non-empty) subsort of Q. Given any set $E = \{(t_{1,1}, t_{1,2}), \ldots, (t_{n,1}, t_{n,2})\}$, where each $t_{i,j}$ $(i = 1..n, j = 1..2)$ is a well-sorted term, a *well-sorted most general unifier* (well-sorted mgu) of E is a well-sorted substitution that is an mgu of E. It is important that in comparison to mgu in unsorted logic (i.e., predicate logic without sorts), mgu in OSL can include new weakened variables of sorts which are subsorts of the sorts of unified terms. For example, assume that $E = \{(x, y)\}$, $x \in \mathbf{V}_{Q_1}$, $y \in \mathbf{V}_{Q_2}$ and the g.l.b. of $\{Q_1, Q_2\}$ is a non-empty sort Q_3. Then, $\mu = [x/z, y/z]$ (x is substituted by z, y is substituted by z) for some new variable $z \in \mathbf{V}_{Q_3}$ is a well-sorted mgu of E. Well-sorted mgu neither always exists nor it is unique. However, it is proved that the well-sorted mgu of unifiable sorted terms is unique up to variable renaming when the sort hierarchy of $\mathcal{T}$ is a meet semi-lattice (Walther 1987).

The semantics of OSL is defined similar to unsorted logic. Note that the definition of interpretations for well-sorted terms and formulas is the same as in unsorted logic, but the semantics is not defined for ill-sorted terms and formulas. For any well-sorted formula ϕ, a $\mathcal{T}$-interpretation $\mathbb{I} = \langle \mathcal{M}, I \rangle$ is a tuple for a structure $\mathcal{M}$ and an assignment I from the set of free variables to the universe $\mathbf{U}$ of $\mathcal{M}$, s.t. it satisfies the following conditions: (1) For each sort Q, $Q^{\mathbb{I}}$ is a subset of the whole universe $\mathbf{U}$. In particular, $\top^{\mathbb{I}} = \mathbf{U}$, $\bot^{\mathbb{I}} = \emptyset$, and $Q_1^{\mathbb{I}} \subseteq Q_2^{\mathbb{I}}$ for any $Q_1 \leq_{\mathcal{T}} Q_2$. (2) For any predicate declaration $P : \vec{Q}_{1..n}$, $P^{\mathbb{I}} \subseteq Q_1^{\mathbb{I}} \times \cdots \times Q_n^{\mathbb{I}}$ is a relation in $\mathcal{M}$. (3) For any function declaration $f : \vec{Q}_{1..n} \rightarrow Q$, $f^{\mathbb{I}} : Q_1^{\mathbb{I}} \times \cdots \times Q_n^{\mathbb{I}} \rightarrow Q^{\mathbb{I}}$ is a function in $\mathcal{M}$. (4) $x^{\mathbb{I}} = I(x)$ is in $Q^{\mathbb{I}}$ for any variable $x \in \mathbf{V}_Q$, $c^{\mathbb{I}} \in Q^{\mathbb{I}}$ for any constant declaration $c : Q$, and $(f(\vec{t}_{1..n}))^{\mathbb{I}} \overset{def}{=} f^{\mathbb{I}}(t_1^{\mathbb{I}}, \ldots, t_n^{\mathbb{I}})$ for any well-sorted term $f(\vec{t}_{1..n})$. $\mathbb{I}$ is not defined for ill-sorted terms and formulas. (5) If $\mathcal{T}$ includes a declaration for equality symbol "=", then $=^{\mathbb{I}}$ must be defined as set $\{(d, d) \mid d \in \mathbf{U}\}$, i.e., the equality symbol is interpreted by the identity relation on the whole universe. For any sort theory $\mathcal{T}$ and a well-sorted formula ϕ, a structure $\mathcal{M}$ is a *$\mathcal{T}$-model* of ϕ, written as $\mathcal{M} \models^{\mathrm{os}}_{\mathcal{T}} \phi$ iff for every $\mathcal{T}$-interpretation $\mathbb{I} = \langle \mathcal{M}, I \rangle$, $\mathbb{I}$ satisfies ϕ. In particular, when ϕ is a sentence, this does not depend on any variable assignment and $\mathbb{I} = \mathcal{M}$. Moreover, we say that a $\mathcal{T}$-interpretation $\mathbb{I} = \langle \mathcal{M}, I \rangle$ satisfies ϕ, written as $\mathbb{I} \models^{\mathrm{os}}_{\mathcal{T}} \phi$, if the following conditions (1-7) hold: (1) $\mathbb{I} \models^{\mathrm{os}}_{\mathcal{T}} P(\vec{t}_{1..n})$ iff $(t_1^{\mathbb{I}}, \ldots, t_n^{\mathbb{I}}) \in P^{\mathbb{I}}$. (2) $\mathbb{I} \models^{\mathrm{os}}_{\mathcal{T}} \neg\phi$ iff $\mathbb{I} \models^{\mathrm{os}}_{\mathcal{T}} \phi$ does not hold. (3) $\mathbb{I} \models^{\mathrm{os}}_{\mathcal{T}} \phi_1 \wedge \phi_2$ iff $\mathbb{I} \models^{\mathrm{os}}_{\mathcal{T}} \phi_1$ and $\mathbb{I} \models^{\mathrm{os}}_{\mathcal{T}} \phi_2$. (4) $\mathbb{I} \models^{\mathrm{os}}_{\mathcal{T}} \phi_1 \vee \phi_2$ iff $\mathbb{I} \models^{\mathrm{os}}_{\mathcal{T}} \phi_1$ or $\mathbb{I} \models^{\mathrm{os}}_{\mathcal{T}} \phi_2$. (5) $\mathbb{I} \models^{\mathrm{os}}_{\mathcal{T}} \phi_1 \supset \phi_2$ iff $\mathbb{I} \models^{\mathrm{os}}_{\mathcal{T}} \neg\phi_1 \vee \phi_2$. (6) $\mathbb{I} \models^{\mathrm{os}}_{\mathcal{T}} \forall x : Q.\phi$ iff for every $d \in Q^{\mathbb{I}}$, $\mathbb{I} \models^{\mathrm{os}}_{\mathcal{T}} \phi[x/d]$, where $\phi[x/o]$ represent the formula obtained by substituting x with o. (7) $\mathbb{I} \models^{\mathrm{os}}_{\mathcal{T}} \exists x : Q.\phi$ iff there is some $d \in Q^{\mathbb{I}}$ s.t. $\mathbb{I} \models^{\mathrm{os}}_{\mathcal{T}} \phi[x/d]$. Given a sort theory $\mathcal{T}$ as the background, a theory Φ including well-sorted sentences only satisfies a well-sorted sentence ϕ, written as $\Phi \models^{\mathrm{os}}_{\mathcal{T}} \phi$, iff every model of Φ is a model of ϕ.

Note that we follow traditional approaches to sorted reasoning, where sort symbols must not occur as predicates in the formulas. Alternative approaches, called hybrid, allow to mix sort symbols with application specific predicates (see (Weidenbach 1996; Cohn 1989; Bierle *et al.* 1992)).

Due to the space limitations, we skip the background of the situation calculus. Details can be found in (Reiter 2001) and we refer to this language as Reiter's situation calculus below. Note that in this paper, we use $\models^{\mathrm{os}}_{\mathcal{T}}$ to represent the logical entailment wrt a sort theory $\mathcal{T}$ in order-sorted logic, $\models^{\mathrm{ms}}$ to represent the logical entailment in Reiter's situation calculus (a many-sorted logic with one standard sort *Object*), and $\models^{\mathrm{fo}}$ to represent the logical entailment in unsorted predicate logic.

An Order-Sorted Situation Calculus

In this paper, we consider a modified situation calculus based on order-sorted logic, called *order-sorted situation calculus* and denoted as $\mathcal{L}^{OS}$ below. $\mathcal{L}^{OS}$ includes a set of sorts **Sort** $=$ **Sort**$_{obj} \cup \{\top, \bot, Act, Sit\}$, where $\top$ represents the whole universe, $\bot$ is the empty sort, Act is the sort for all actions, Sit is the sort for all situations, and **Sort**$_{obj}$ is a set of sub-sorts of $Object$ including sort $Object$ itself. We assume that for every sort (except $\bot$) there is at least one ground term (constant) of this sort to avoid the problem with "empty sorts" (Goguen & Meseguer 1987). Moreover, the number of individual variable symbols of each sort in **Sort** is infinitely countable. For the sake of simplicity, we do not consider functional fluents here.

In the following, we will define *order-sorted basic action theories* (order-sorted BATs) and consider dynamical systems that can be described using such order-sorted BATs. An order-sorted BAT $\mathcal{D} = (\mathcal{T}_{\mathcal{D}}, \mathbf{D})$ includes the following two parts of theories.

- $\mathcal{T}_{\mathcal{D}}$ is a sort theory based on a finite set of sorts $\mathbf{Q}_{\mathcal{D}}$ s.t. $\mathbf{Q}_{\mathcal{D}} \subseteq$ **Sort** and $\{\bot, \top, Object, Act, Sit\} \subseteq \mathbf{Q}_{\mathcal{D}}$. Moreover, the sort theory includes the following declarations for finitely many predicates and functions:

1. Subsort declarations of the form $Q_1 \leq Q_2$ for $Q_1, Q_2 \in \mathbf{Q}_{\mathcal{D}} - \{\top, \mathit{Act}, \mathit{Sit}\}$, and subsort declarations: $\mathit{Object} \leq \top$, $\mathit{Act} \leq \top$, $\mathit{Sit} \leq \top$. $\bot \leq \mathit{Act}$, $\bot \leq \mathit{Sit}$. Here, we only consider those sort theories whose sort hierarchies are meet semilattices.

2. One and only one predicate declaration of the form $F : \vec{Q}_{1..n}$ for each n-ary relational fluent F in the system, where $Q_i \leq_{\mathcal{T}} \mathit{Object}$ and $Q_i \neq \bot$ for $i = 1..(n-1)$, and Q_n is Sit.

3. One and only one predicate declaration for the special predicate Poss, that is, $\mathit{Poss} : \mathit{Act} \times \mathit{Sit}$.

4. One and only one predicate declaration of the form $P : \vec{Q}_{1..n}$ for each n-ary situation independent predicate P in the system, where $Q_i \leq_{\mathcal{T}} \mathit{Object}$ and $Q_i \neq \bot$ for $i = 1..n$.

5. A special declaration for equality symbol $=: \ \top \times \top$.

6. One and only one function declaration of the form $A : \vec{Q}_{1..n} \rightarrow \mathit{Act}$ for each n-ary action function A in the system, where $Q_i \leq_{\mathcal{T}} \mathit{Object}$ and $Q_i \neq \bot$ for $i = 1..n$. Note that, when $n = 0$, the declaration is of form $A : \mathit{Act}$ for constant action function A.

7. One and only one function declaration of the form $f : \vec{Q}_{1..n} \rightarrow Q_{n+1}$ for each n-ary ($n \geq 0$) situation independent function f (other than action functions), where each $Q_i \leq_{\mathcal{T}} \mathit{Object}$ and $Q_i \neq \bot$ for each $i = 1..(n+1)$. Note that, when $n = 0$, it is a function declaration for a constant, denoted as $c : Q$ for constant c of sort Q.

8. One and only one function declaration $\mathit{do} : \mathit{Act} \times \mathit{Sit} \rightarrow \mathit{Sit}$, and $S_0 : \mathit{Sit}$ for the initial situation S_0.

• **D** is a set of axioms represented using well-sorted sentences wrt $\mathcal{T}_{\mathcal{D}}$, which includes the following subsets of axioms.

1. Foundational axioms Σ for situations, which are the same as those in (Reiter 2001).

2. A set $\mathcal{D}_{una}$ of unique name axioms for actions: for any two distinct action function symbols A and B with declarations $A : \vec{Q}_{1..n} \rightarrow \mathit{Act}$ and $B : \vec{Q}'_{1..m} \rightarrow \mathit{Act}$, we have

$$(\forall \vec{x}_{1..n} : \vec{Q}_{1..n}, \vec{y}_{1..m} : \vec{Q}'_{1..m}).\, A(\vec{x}_{1..n}) \neq B(\vec{y}_{1..m})$$

Moreover, for each action function symbol A, we have

$$(\forall \vec{x}_{1..n} : \vec{Q}_{1..n}, \vec{y}_{1..n} : \vec{Q}_{1..n}).\, A(\vec{x}_{1..n}) = A(\vec{y}_{1..n}) \supset \bigwedge_{i=1}^{n} x_i = y_i$$

3. The initial theory $\mathcal{D}_{S_0}$, which includes well-sorted (first-order) sentences that are uniform in S_0. In particular, it includes the unique name axioms for object terms, object constants and/or functional terms: (1) for any two distinct situation-independent function symbols (including constants) f_1 and f_2, we have $\forall \vec{x}_{1..n} : \vec{Q}_{1..n}. \forall \vec{y}_{1..m} : \vec{Q}'_{1..m} f_1(\vec{x}_{1..n}) \neq f_2(\vec{y}_{1..m})$, where the functional declarations for f_1 and f_2 are $f_1 : \vec{Q}_{1..n} \rightarrow Q_{n+1}$ ($n \geq 0$) and $f_2 : \vec{Q}'_{1..m} \rightarrow Q'_{m+1}$ ($m \geq 0$); (2) for each situation-independent function f, we have $f(\vec{x}) = f(\vec{y}) \supset \wedge_{i=1}^{n} x_i = y_i$, where the functional declaration for f is $f : \vec{Q}_{1..n} \rightarrow Q_{n+1}$ ($n \geq 1$). $\mathcal{D}_{S_0}$ can also include additional constraints that relate to sorts. For instance, there can be finitely many *axioms of disjointness for basic sorts* of the form $\forall x : Q_i. \forall y : Q_j.(x \neq y)$ for any two disjoint *basic sorts* Q_i and Q_j, where a sort Q is considered basic if there is no sort $Q' \neq \bot$, such that $Q' \leq Q$.

4. A set $\mathcal{D}_{ap}$ of precondition axioms for actions represented using well-sorted formulas: for each action symbol A, whose sort declaration is $A : \vec{Q}_{1..n} \rightarrow \mathit{Act}$, its precondition axiom is of the form

$$(\forall \vec{x}_{1..n} : \vec{Q}_{1..n}, s : \mathit{Sit}). \mathit{Poss}(A(\vec{x}_{1..n}), s) \equiv \phi_A(\vec{x}_{1..n}, s), \quad (1)$$

where $\phi_A(\vec{x}_{1..n}, s)$ is a well-sorted formula uniform in s, whose free variables are at most among $\vec{x}_{1..n}$ and s.

5. A set $\mathcal{D}_{ss}$ of successor state axioms (SSAs) for fluents represented using well-sorted formulas: for each fluent F with declaration $F : \vec{Q}_{1..n} \times \mathit{Sit}$, its SSA is of the form

$$\begin{aligned} &(\forall \vec{x}_{1..n} : \vec{Q}_{1..n}, a : \mathit{Act}, s : \mathit{Sit}). \\ &\quad F(\vec{x}_{1..n}, \mathit{do}(a, s)) \equiv \psi_F(\vec{x}_{1..n}, a, s), \end{aligned} \quad (2)$$

where $\psi_F(\vec{x}_{1..n}, a, s)$ is a well-sorted formula uniform in s, whose free variables are at most among $\vec{x}_{1..n}$ and a, s.

Here is a simple example of an order-sorted BAT.

Example 1 (Transport Logistics) We present an order-sorted BAT $\mathcal{D}$ of a simplified example of logistics. $\mathcal{T}_{\mathcal{D}}$ includes following subsort declarations:

$$\mathit{MovObj} \leq \mathit{Object},\ \bot \leq \mathit{City},\ \bot \leq \mathit{Box},\ \bot \leq \mathit{Truck},$$
$$\mathit{Truck} \leq \mathit{MovObj},\ \mathit{City} \leq \mathit{Object},\ \mathit{Box} \leq \mathit{MovObj},$$

where MovObj is the sort of movable objects, and other sorts are self-explanatory. The predicate declarations are

$$\mathit{InCity} : \mathit{MovObj} \times \mathit{City} \times \mathit{Sit},\ \ \mathit{On} : \mathit{Box} \times \mathit{Truck} \times \mathit{Sit}$$

for the fluents $\mathit{InCity}(o, l, s)$ and $\mathit{On}(o, t, s)$. The function declarations for actions $\mathit{load}(b, t)$, $\mathit{unload}(b, t)$ and $\mathit{drive}(t, c_1, c_2)$ are obvious. For instance,

$$\mathit{drive} : \mathit{Truck} \times \mathit{City} \times \mathit{City} \rightarrow \mathit{Act}$$

Besides $S_0 : \mathit{Sit}$, the constant declarations may include:

$$\begin{array}{lll} B_1 : \mathit{Box}, & B_2 : \mathit{Box}, & T_1 : \mathit{Truck}, \\ T_2 : \mathit{Truck}, & \mathit{Pasadena} : \mathit{City}, & \mathit{Boston} : \mathit{City}. \end{array}$$

Axioms in $\mathcal{D}_{S_0}$ can be:

$$\exists x : \mathit{Box}.\, \mathit{InCity}(x, \mathit{Boston}, S_0),$$
$$(\forall x : \mathit{Box}, t : \mathit{Truck}).\, \neg \mathit{On}(x, t, S_0),$$
$$\mathit{InCity}(T_1, \mathit{Boston}, S_0) \vee \mathit{InCity}(T_2, \mathit{Boston}, S_0).$$

As an example, the precondition axiom for load is:

$$\begin{aligned} &(\forall x : \mathit{Box}, t : \mathit{Truck}, s : \mathit{Sit}).\, \mathit{Poss}(\mathit{load}(x, t), s) \equiv \\ &\quad \neg \mathit{On}(x, t, s) \wedge \exists y : \mathit{City}.\mathit{InCity}(x, y, s) \wedge \mathit{InCity}(t, y, s), \end{aligned}$$

and the preconditions for unload and drive are obvious.
As an example, the SSA of fluent InCity is:

$$\begin{aligned} &(\forall d : \mathit{MovObj}, c : \mathit{City}, a : \mathit{Act}, s : \mathit{Sit}). \\ &\ \mathit{InCity}(d, c, \mathit{do}(a, s)) \equiv (\exists t : \mathit{Truck}, c_1 : \mathit{City}). \\ &\ a = \mathit{drive}(t, c_1, c) \wedge (d = t \vee \exists b : \mathit{Box}.b = d \wedge \mathit{On}(b, t, s))) \vee \\ &\ \mathit{InCity}(d, c, s) \wedge \neg(\exists t : \mathit{Truck}, c_1 : \mathit{City}.a = \mathit{drive}(t, c, c_1) \\ &\ \wedge(d = t \vee \exists b : \mathit{Box}.b = d \wedge \mathit{On}(b, t, s))), \end{aligned}$$

and the SSA of fluent On is obvious.

Order-Sorted Regression and Reasoning

We now consider the central reasoning mechanism in the order-sorted situation calculus. The definition of a regressable formula of $\mathcal{L}^{OS}$ is the same as the definition of a regressable formula of $\mathcal{L}_{sc}$ except that instead of being stated for a formula in $\mathcal{L}_{sc}$, it is formulated for a well-sorted formula in $\mathcal{L}^{OS}$.

A formula W of $\mathcal{L}^{OS}$ is *regressable* (wrt an order-sorted BAT $\mathcal{D}$) iff (1) W is a well-sorted first-order formula wrt $\mathcal{T}_{\mathcal{D}}$; (2) every term of sort Sit in W starts from S_0 and has the syntactic form $\mathit{do}([\alpha_1, \cdots, \alpha_n], S_0)$, where each α_i is of sort Act; (3) for every atom of the form $\mathit{Poss}(\alpha, \sigma)$ in W, α has the syntactic form $A(\vec{t}_{1..n})$ for some n-ary action function symbol A; and (4) W does not quantify over situations, and does not mention the relation symbols "$\sqsubset$" or "$=$" between terms of sort Sit. A *query* is a regressable sentence.

Example 2 Consider the BAT $\mathcal{D}$ from Example 1. Let W be $\exists d\!:\!Box.\, d\!=\!Boston \wedge On(d, T_1, do(load(B_1, T_1), S_0))$
W is a (well-sorted) regressable sentence (wrt $\mathcal{D}$); while

$$On(Boston, T_1, do(load(B_1, T_1), S_0))$$

is ill-sorted and therefore is not regressable.

The regression operator $\mathcal{R}^{os}$ in $\mathcal{L}^{OS}$ is defined recursively similar to the regression operator in (Reiter 2001). Moreover, we would like to take advantages of the sort theory during regression: when there is no well-sorted mgu for equalities between terms that occur in a conjunctive sub-formula of a query, this sub-formula is logically equivalent to false and it should not be regressed any further. We will see that this key idea helps eliminate useless sub-trees of a regression tree. In what follows, $\vec{t}$ and $\vec{\tau}$ are tuples of terms, α and α' are terms of sort Act, σ and σ' are terms of sort Sit, and W is a regressable formula of $\mathcal{L}^{OS}$.

1. If W is a non-atomic formula and is of the form $\neg W_1$, $W_1 \vee W_2$, $(\exists v : Q).W_1$ or $(\forall v : Q).W_1$, for some regressable formulas W_1, W_2 in $\mathcal{L}^{OS}$, then

$$\mathcal{R}^{os}[\circ W_1] = \circ\mathcal{R}^{os}[W_1] \text{ for constructor } \circ \in \{\neg, (\exists x\!:\!Q), (\forall x\!:\!Q)\}$$
$$\mathcal{R}^{os}[W_1 \vee W_2] = \mathcal{R}^{os}[W_1] \vee \mathcal{R}^{os}[W_2].$$

2. Else, if W is a non-atomic formula, W is not of the form $\neg W_1$, $W_1 \vee W_2$, $(\exists v : Q)W_1$ or $(\forall v : Q)W_1$, but of the form $W_1 \wedge W_2 \wedge \cdots \wedge W_n$ ($n \geq 2$), where each W_i ($i = 1..n$) is not of the form $W_{i,1} \wedge W_{i,2}$ for some sub-formulas $W_{i,1}, W_{i,2}$ in W_i. After using commutative law for $\wedge$, without loss of generality, there are two sub-cases:

2(a) Suppose that for some j, $j = 1..n$, each W_i ($i = 1..j$) is of the form $t_{i,1} = t_{i,2}$ for some (well-sorted) terms $t_{i,1}, t_{i,2}$, and none of W_k, $k = (j+1)..n$, is an equality between terms. In particular, when $j = n$, $\bigwedge_{k=j+1}^{n} W_k \overset{def}{=} true$. Then,

$$\mathcal{R}^{os}[W] = \begin{cases} W_1 \wedge W_2 \wedge \cdots \wedge W_j \wedge \mathcal{R}^{os}[W_0'] \\ \quad \text{if there is a well-sorted mgu } \mu \\ \qquad \text{for } \{\langle t_{i,1}, t_{i,2}\rangle \mid i = 1..j\}; \\ false \qquad\qquad \text{otherwise.} \end{cases}$$

Here, W_0' is a new formula obtained by applying mgu μ to $\bigwedge_{k=j+1}^{n} W_k$ and it is existentially-quantified at front for every newly introduced sort weakened variable in μ. Moreover, note that based on the assumption that we consider meet semi-lattice sort hierarchies only, such mgu is unique if it exists.

2(b) Otherwise, $\mathcal{R}^{os}[W] = \mathcal{R}^{os}[W_1] \wedge \cdots \wedge \mathcal{R}^{os}[W_n]$.

3. Otherwise, W is atomic. There are four sub-cases.

3(a) Suppose that W is of the form $Poss(A(\vec{t}), \sigma)$ for an action term $A(\vec{t})$ and a situation term σ, and the action precondition axiom for A is of the form (1). Without loss of generality, assume that all variables in Axiom (1) have had been renamed (with variables of the same sorts) to be distinct from the free variables (if any) of W. Then,

$$\mathcal{R}^{os}[W] = \mathcal{R}^{os}[\phi_A(\vec{t}, \sigma)].$$

3(b) Suppose that W is of the form $F(\vec{t}, do(\alpha, \sigma))$ for some relational fluent F. Let F's SSA be of the form (2). Without loss of generality, assume that all variables in Axiom (2) have had been renamed (with variables of the same sorts) to be distinct from the free variables (if any) of W. Then, $\quad \mathcal{R}^{os}[W] = \mathcal{R}^{os}[\psi_F(\vec{t}, \alpha, \sigma)]$.

3(c) Suppose that atom W is of the form $t_1 = t_2$. for some well-sorted terms t_1, t_2. Then,

$$\mathcal{R}^{os}[W] = \begin{cases} W & \text{if there is a well-sorted mgu } \mu \\ & \qquad \text{for } \langle t_1, t_2\rangle; \\ false & \qquad \text{otherwise.} \end{cases}$$

3(d) Otherwise, if atom W has S_0 as its only situation term, then $\quad \mathcal{R}^{os}[W] = W$.

Notice that although the definition seems to depend on syntactic form of a formula, we prove below that for any regressable formulas W_1 and W_2 in $\mathcal{L}^{OS}$ that are logically equivalent, their regressed results are still equivalent wrt $\mathcal{D}$ (See Corollary 1). Here are some examples.

Example 3 Consider the order-sorted BAT $\mathcal{D}$ from Example 1 and the query W from Example 2. Then, it is easy to see that $\mathcal{R}^{os}[W] = false$, since there is no well-sorted mgu for $(d, Boston)$, where $d\!:\!Box$. Now, let W_1 be

$$\neg\forall d\!:\!Box.\, d \neq Boston \vee \neg On(d, T_1, do(load(B_1, T_1), S_0)).$$

W_1 is a sentence that is equivalent to W. It is easy to check that $\mathcal{R}^{os}[W_1]$ is a formula equivalent to $false$ (wrt $\mathcal{D}$).

Given an order-sorted BAT $\mathcal{D} = (\mathcal{T}_{\mathcal{D}}, \mathbf{D})$ and the order-sorted regression operator defined above, to show the correctness of the newly defined regression operator, we prove the following theorems similar to that of in (Reiter 2001).

Theorem 1 *If W is a regressable formula wrt $\mathcal{D}$, then $\mathcal{R}^{os}[W]$ is a well-sorted $\mathcal{L}^{OS}$ formula (including $false$) that is uniform in S_0. Moreover, $\mathbf{D} \models^{\text{os}}_{\mathcal{T}_{\mathcal{D}}} W \equiv \mathcal{R}^{os}[W]$.*

Theorem 2 *If W is a regressable formula wrt $\mathcal{D}$, then*

$$\mathbf{D} \models^{\text{os}}_{\mathcal{T}_{\mathcal{D}}} W \;\; \textit{iff} \;\; \mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models^{\text{os}}_{\mathcal{T}_{\mathcal{D}}} \mathcal{R}^{os}[W].$$

Hence, to reason whether $\mathbf{D} \models^{\text{os}}_{\mathcal{T}_{\mathcal{D}}} W$ is the same as to compute $\mathcal{R}^{os}[W]$ first and then to reason whether $\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models^{\text{os}}_{\mathcal{T}_{\mathcal{D}}} \mathcal{R}^{os}[W]$. Besides, according to Theorem 1, it is easy to see that the following consequence holds.

Corollary 1 *If W_1 and W_2 are regressable formulas in $\mathcal{L}^{OS}$ s.t. $\models^{\text{os}}_{\mathcal{T}_{\mathcal{D}}} W_1 \equiv W_2$, then $\mathbf{D} \models^{\text{os}}_{\mathcal{T}_{\mathcal{D}}} \mathcal{R}^{os}[W_1] \equiv \mathcal{R}^{os}[W_2]$.*

Intuitively, Corollary 1 states that the regressed results of two logically equivalent regressable formulas (possibly having different syntactic forms only) are still equivalent.

Order-Sorted Situation Calculus v.s. Reiter's Situation Calculus

Although BATs and regressable formulas in $\mathcal{L}^{OS}$ are based on OSL, they can be related to BATs and regressable formulas in Reiter's situation calculus as stated in Theorem 3.

Theorem 3 (Soundness) *For any BAT $\mathcal{D}$ and any query W in order-sorted situation calculus $\mathcal{L}^{OS}$, there exists a corresponding BAT $\mathcal{D}'$ and a corresponding query W' in Reiter's situation calculus s.t.*

$$\mathbf{D} \models^{\text{os}}_{\mathcal{T}_{\mathcal{D}}} W \;\; \textit{iff} \;\; \mathcal{D}' \models^{\text{ms}} W'.$$

Intuitively, we would like to show that the order-sorted situation calculus $\mathcal{L}^{OS}$ is correct, or *sound*, in the sense that for any query in $\mathcal{L}^{OS}$ that can be answered in its background BAT in $\mathcal{L}^{OS}$, we always can find a way to represent the BAT and the query in Reiter's situation calculus $\mathcal{L}_{sc}$ s.t. the corresponding query in $\mathcal{L}_{sc}$ can be answered wrt the corresponding BAT in $\mathcal{L}_{sc}$.

It is hard to prove Theorem 3 directly. Inspired by the *standard relativization* of OSL to unsorted (first-order) logic, our general idea of proving Theorem 3 is as follows. In Step 1, we prove that there is an unsorted theory $\mathcal{D}''$ (via *strong relativization*) and an unsorted first-order sentence W'' (via *relativization*) s.t. $\mathbf{D} \models^{\mathrm{os}}_{\mathcal{T}_\mathcal{D}} W$ iff $\mathcal{D}'' \models^{\mathrm{fo}} W''$. In Step 2, we construct a BAT $\mathcal{D}'$ (called the *corresponding Reiter's BAT of* $\mathcal{D}$ below) and a regressable formula W' (called the *translation of* W below) in Reiter's situation calculus, s.t. $\mathcal{D}' \models^{\mathrm{ms}} W'$ iff $\mathcal{D}''' \models^{\mathrm{fo}} W'''$, for some unsorted theory $\mathcal{D}'''$ (via standard relativization) and sentence W''' (via relativization). Finally, in Step 3, we show that $\mathcal{D}''' \models^{\mathrm{fo}} W'''$ iff $\mathcal{D}'' \models^{\mathrm{fo}} W''$.

$$\begin{tikzcd}
\mathbf{D} \models^{\mathrm{os}}_{\mathcal{T}_\mathcal{D}} W \arrow[r, Leftrightarrow, "\text{(Step 1)}"] & \mathcal{D}'' \models^{\mathrm{fo}} W'' \arrow[d, Leftrightarrow, "\text{(Step 3)}"] \\
\mathcal{D}' \models^{\mathrm{ms}} W' \arrow[r, Leftrightarrow, "\text{(Step 2)}"] & \mathcal{D}''' \models^{\mathrm{fo}} W'''
\end{tikzcd}$$

Fig 1. Diagram of the Outline for Proving Theorem 3

To prove Theorem 3, we first define some concepts and prove Lemma 1 for later convenience. First, for any sort Q in the language of $\mathcal{L}^{OS}$, we introduce a unary predicate $Q(x)$, which will be true iff x is of sort Q in $\mathcal{L}^{OS}$.

Definition 1 For any well-sorted formula ϕ in $\mathcal{L}^{OS}$, $rel(\phi)$, a *relativization* of ϕ, is an unsorted formula defined as:

For every atom $P(\vec{t})$, $rel(P(\vec{t})) \overset{def}{=} P(\vec{t})$; $rel(\neg\phi) \overset{def}{=} \neg rel(\phi)$;
$rel(\phi \circ \psi) \overset{def}{=} rel(\phi) \circ rel(\psi)$ for $\circ \in \{\wedge, \vee, \supset\}$;
$rel((\forall x\!:\!Q)\phi) \overset{def}{=} (\forall y)[Q(y) \supset rel(\phi[x/y])]$;
$rel((\exists x\!:\!Q)\phi) \overset{def}{=} (\exists y)[Q(y) \wedge rel(\phi[x/y])]$.

Moreover, for any set Set of well-sorted formulas,
$$rel(Set) = \{rel(\phi) \mid \phi \in Set\}.$$

Note that all formulas in $\mathcal{L}_{sc}$ are well-sorted wrt the sort theory of $\mathcal{L}_{sc}$. Hence, the definition of rel can also be applied to any formula or a set of formulas in Reiter's situation calculus.

Definition 2 For any sort theory $\mathcal{T}_\mathcal{D}$ in $\mathcal{L}^{OS}$, *the set of bridge axioms* of $\mathcal{T}_\mathcal{D}$, $BA(\mathcal{T}_\mathcal{D})$, is a set of the following formulas:
(a) $(\forall x).\, Q_2(x) \supset Q_1(x)$ for each $Q_2 \leq Q_1 \in \mathcal{T}_\mathcal{D}$;
(b) $Q(c)$ for each $c\!:\!Q \in \mathcal{T}_\mathcal{D}$;
(c) $(\forall \vec{x}_{1..n}).\, \bigwedge_{i=1}^{n} Q_i(x_i) \supset Q(f(\vec{x}_{1..n}))$ for each $f : \vec{Q}_{1..n} \rightarrow Q \in \mathcal{T}_\mathcal{D}$.

Moreover, let $Sorted(x)$ be an auxiliary predicate that does not appear in $\mathcal{D}$: it is a purely technical device used for proving Theorem 3. The set of *strong bridge axioms* of $\mathcal{T}_\mathcal{D}$, $SBA(\mathcal{T}_\mathcal{D})$, is also a set of unsorted axioms $BA(\mathcal{T}_\mathcal{D}) \cup sba(\mathcal{T}_\mathcal{D})$, where $sba(\mathcal{T}_\mathcal{D})$ includes the following axioms:
(d) $(\forall \vec{x}_{1..n}).P(\vec{x}_{1..n}) \supset \bigwedge_{i=1}^{n} Q_i(x_i) \wedge Sorted(x_i)$ for each $P\!:\!\vec{Q}_{1..n} \in \mathcal{T}_\mathcal{D}$;
(e) $(\forall \vec{x}_{1..n}).Q(f(\vec{x}_{1..n})) \wedge Sorted(f(\vec{x}_{1..n})) \supset \bigwedge_{i=1}^{n}(Q_i(x_i) \wedge Sorted(x_i))$ for each $f\!:\!\vec{Q}_{1..n} \rightarrow Q \in \mathcal{T}_\mathcal{D}$.

Intuitively, $Sorted(t)$ means that term t is well-sorted (wrt $\mathcal{D}$). (a functional term is well-sorted and of its own sort, respectively), then all its arguments should be well-sorted and of the corresponding sorts wrt the predicate declaration (the function declaration, respectively). Note that although $Sorted$ may satisfy other characterizing axioms than axioms in (d) and (e) according to its intuitive meaning, but adding axioms in (d) and (e) to the strong relativization theory of $\mathcal{D}$ defined below is enough for us to prove Theorem 3.

Definition 3 For any order-sorted BAT $\mathcal{D}$ in $\mathcal{L}^{OS}$, *the strong relativization of* $\mathcal{D}$, an unsorted theory, is defined as
$$REL_S(\mathcal{D}) \overset{def}{=} rel(\mathbf{D}) \cup SBA(\mathcal{T}_\mathcal{D}).$$
Consider any BAT $\mathcal{D}_1$ in Reiter's situation calculus $\mathcal{L}_{sc}$, which has a finite set $\mathcal{T}_{\mathcal{D}_1}$ of function declarations and predicate declarations for all predicates and functions appeared in $\mathcal{D}_1$. The *standard relativization of* $\mathcal{D}_1$, an unsorted theory, is defined as
$$REL(\mathcal{D}_1) \overset{def}{=} rel(\mathcal{D}_1) \cup BA(\mathcal{T}_{\mathcal{D}_1}).$$

The reasons for differences between the two cases in Def. 3 are that (1) we include the sort theory in each BAT of order-sorted situation calculus, while Reiter's situation calculus mentions sort declarations generally in the signature of $\mathcal{L}_{sc}$, and (2) we need strong relativization for order-sorted BATs and only need standard relativization for Reiter's BATs to prove Theorem 3. In comparison to the standard relativization, the strong relativization adds additional axioms of the form (d) and (e) in Def. 2. They are based on the sort theory that includes one and only one declaration for each predicate P or for each function f, respectively. We can also prove a relativization theorem as follows for the strong relativization similar to the Sort Theorem proved in (Walther 1987) and/or the relativization theorem proved in (Schmidt-Schauβ 1989).

Lemma 1 *Consider any regressable formula W with a background BAT $\mathcal{D}$ in order-sorted situation calculus $\mathcal{L}^{OS}$. Then,*
$$\mathbf{D} \models^{\mathrm{os}}_{\mathcal{T}_\mathcal{D}} W \textit{ iff } REL_S(\mathcal{D}) \models^{\mathrm{fo}} rel(W).$$

We therefore can prove Step 1 in Fig. 1 using Lemma 1. Because Reiter's situation calculus is a many-sorted logical language with special formats for precondition axioms and SSAs, we cannot use rel to relate $\mathcal{D}$ in $\mathcal{L}^{OS}$ with a Reiter's BAT directly. It is also the reason why strong relativization is introduced. To construct a Reiter's BAT $\mathcal{D}'$ and a regressable formula W' that satisfy the theorem, we first define another translation function $tr(W)$ as follows.

Definition 4 Consider any well-sorted formula ϕ in $\mathcal{L}^{OS}$. A *translation* of ϕ to a (well-sorted) sentence in Reiter's situation calculus, denoted as $tr(\phi)$, is defined recursively as follows:

For every atom $P(\vec{t})$, $tr(P(\vec{t})) \overset{def}{=} P(\vec{t})$; $tr(\neg\phi) \overset{def}{=} \neg tr(\phi)$;
$tr((\exists x\!:\!\bot)\phi) \overset{def}{=} false$; $tr((\forall x\!:\!Q)\phi) \overset{def}{=} \neg tr((\exists x\!:\!Q.\neg\phi))$;
$tr((\exists x\!:\!Q)\phi) \overset{def}{=} (\exists x\!:\!Q)tr(\phi)$, if $Q \in \{Object, Act, Sit\}$.
$tr((\exists x\!:\!\top)\phi) \overset{def}{=} (\exists x\!:\!Object)tr(\phi) \vee (\exists x\!:\!Act)tr(\phi) \vee$
$(\exists x\!:\!Sit)tr(\phi)$;
$tr((\exists x\!:\!Q)\phi) \overset{def}{=} (\exists y\!:\!Object)[Q(y) \wedge tr(\phi(x/y))]$,

$$\text{if } Q \not\in \{\top, \bot, Object, Act, Sit\};$$
$$tr(\phi \circ \psi) \overset{def}{=} tr(\phi) \circ tr(\psi) \text{ for } \circ \in \{\supset, \wedge, \vee, \supset, \equiv\}.$$

The translation function tr defined above is a mapping from well-sorted formulas wrt the sort theory of some BAT $\mathcal{D}$ (or, wrt $\mathcal{D}$ for simplicity) in $\mathcal{L}^{OS}$ to well-sorted formulas in $\mathcal{L}_{sc}$. Moreover, it is easy to prove by structural induction the following lemma for rel and tr, which will be useful for proving Theorem 3.

Lemma 2 *Consider any well-sorted formula ϕ in $\mathcal{L}^{OS}$. Then, $\models^{\text{fo}} rel(tr(\phi)) \equiv rel(\phi)$.*

Consider any order-sorted BAT $\mathcal{D}$. We construct the *corresponding Reiter's BAT of $\mathcal{D}$*, denoted as $TR(\mathcal{D})$, that will be the Reiter's BAT we are looking for in Theorem 3. Notice that in (Reiter 2001), sorted quantifiers are omitted as a convention, because their sorts are always obvious from context. Hence, when we construct the BAT $TR(\mathcal{D})$ in Reiter's situation calculus below, all free variables are implicitly universally sorted-quantified according to their obvious sorts. The function and predicate declarations are always standard, hence are not mentioned here.

- $TR(\mathcal{D})$ includes the foundational axioms and the set of unique name axioms for action functions in Reiter's situation calculus.
- The initial theory of $TR(\mathcal{D})$, say $\mathcal{D}'_{S_0}$, includes the following axioms. Note that for axioms in items **(3)**–**(5)** below, predicate $Sorted$ is auxiliary wrt $\mathcal{D}$ and each x_i is universally quantified with a default sort $Object$ (Q_i itself, respectively) if $Q_i \leq_{\mathcal{T}} Object$ ($Q_i \not\leq_{\mathcal{T}} Object$, respectively).

1. For any well-sorted sentence $\phi \in \mathcal{D}_{S_0}$, $tr(\phi)$ is in $\mathcal{D}'_{S_0}$.
2. For each declaration $Q_2 \leq Q_1$ in $\mathcal{T}_{\mathcal{D}}$, add an axiom $tr((\forall x : \top).(\exists y_2 : Q_2.x = y_2) \supset (\exists y_1 : Q_1.x = y_1))$.
3. For each declaration $f : \vec{Q}_{1..n} \rightarrow Q$ in $\mathcal{T}_{\mathcal{D}}$ ($n \geq 1$), add an axiom $tr((\forall \vec{x}_{1..n} : \vec{Q}_{1..n}).(\exists y : Q).y = f(\vec{x}_{1..n}))$.
We also add an axiom

$$Q(f(\vec{x}_{1..n})) \wedge Sorted(f(\vec{x}_{1..n})) \supset$$
$$tr((\exists \vec{y}_{1..n} : \vec{Q}_{1..n}). \textstyle\bigwedge_{i=1}^{n}(x_i = y_i \wedge Sorted(x_i)))$$

if $Q \leq_{\mathcal{T}} Object$ and $Q \neq Object$, or add an axiom

$$((\exists y : Q).y = f(\vec{x}_{1..n}) \wedge Sorted(y)) \supset$$
$$tr((\exists \vec{y}_{1..n} : \vec{Q}_{1..n}). \textstyle\bigwedge_{i=1}^{n}(x_i = y_i \wedge Sorted(x_i)))$$

otherwise.
4. For each situation-independent predicate declaration $P : \vec{Q}_{1..n}$, add an axiom
$P(\vec{x}_{1..n}) \supset tr((\exists \vec{y}_{1..n} : \vec{Q}_{1..n}). \bigwedge_{i=1}^{n}(x_i = y_i \wedge Sorted(x_i)))$.

5. For each fluent declaration $F : \vec{Q}_{1..n} \times Sit$, add an axiom
$F(\vec{x}_{1..n}, S_0) \supset tr((\exists \vec{y}_{1..n} : \vec{Q}_{1..n}). \bigwedge_{i=1}^{n}(x_i = y_i \wedge Sorted(x_i)))$.

6. For any constant declaration $c : Q$ where $Q \leq_{\mathcal{T}} Object$ and $Q \neq Object$, add an axiom $Q(c)$. Note that other constant declarations will still be kept in the sort theory of $\mathcal{L}_{sc}$ by default (e.g., $S_0 : Sit$).

- For action $A(\vec{x}_{1..n})$ whose precondition axiom in $\mathcal{D}_{ap}$ has the form Eq. (1), we replace it with a precondition axiom in the format of Reiter's situation calculus:

$$Poss(A(\vec{x}_{1..n}), s) \quad \equiv \quad \phi'_A(\vec{x}_{1..n}, s) \tag{3}$$

where $\phi'_A(\vec{x}_{1..n}, s)$ is a $\mathcal{L}_{sc}$ formula uniform in s, resulting from $tr((\exists \vec{y}_{1..n} : \vec{Q}_{1..n}).(\bigwedge_{i=1}^{n} x_i = y_i) \wedge \phi_A(\vec{y}_{1..n}, s))$. Here, all y_i's are distinct auxiliary variables never appearing in $\phi_A(\vec{x}_{1..n}, s)$.

- For each relational fluent $F(\vec{x}_{1..n}, s)$, whose SSA in $\mathcal{D}_{ss}$ is of the form Eq. (2), we replace it with SSA in the format of Reiter's situation calculus:

$$F(\vec{x}_{1..n}, do(a, s)) \quad \equiv \quad \psi'_F(\vec{x}_{1..n}, a, s) \tag{4}$$

where $\psi'_F(\vec{x}_{1..n}, a, s)$ is a $\mathcal{L}_{sc}$ formula uniform in s, resulting from $tr((\exists \vec{y}_{1..n} : \vec{Q}_{1..n}). \bigwedge_{i=1}^{n} x_i = y_i \wedge \psi_F(\vec{y}_{1..n}, a, s))$. Here, all y_i's are distinct auxiliary variables never appearing in $\psi_F(\vec{x}_{1..n}, s)$.

Let $\mathcal{D}' = TR(\mathcal{D})$, $W' = tr(W)$, we then can prove Theorem 3 by following the ideas presented in Fig. 1. Details are omitted due to the space limitations.

Example 4 Consider the BAT $\mathcal{D}$ from Example 1. The axioms in $TR(\mathcal{D})$ are mostly obvious. Due to the space limitations, we just provide examples of a precondition axiom and an SSA in $TR(\mathcal{D})$:

$$\begin{aligned}
&Poss(load(x,t), s) \equiv Box(x) \wedge Truck(t) \wedge \neg On(x,t,s) \wedge \\
&\qquad (\exists y.City(y) \wedge InCity(x,y,s) \wedge InCity(t,y,s)), \\
&InCity(d,c,do(a,s)) \equiv MovObj(d) \wedge City(c) \wedge \\
&\qquad [(\exists t, c_1.Truck(t) \wedge City(c_1) \wedge a = drive(t, c_1, c) \\
&\qquad \wedge (d = t \vee \exists b.Box(b) \wedge b = d \wedge On(b,t,s))) \\
&\qquad \vee InCity(d,c,s) \wedge \\
&\qquad \neg(\exists t, c_1.Truck(t) \wedge City(c_1) \wedge a = drive(t, c, c_1) \\
&\qquad \wedge (d = t \vee \exists b.Box(b) \wedge b = d \wedge On(b,t,s)))].
\end{aligned}$$

It is important to notice that all queries $\mathcal{L}^{OS}$ have to be well-sorted wrt the given background order-sorted BAT $\mathcal{D}$; while, in general, the queries that can be answered in the corresponding Reiter's BAT of $\mathcal{D}$ are not necessarily well-sorted wrt $\mathcal{D}$. Below, Theorem 4 shows that for any query that can be answered in $TR(\mathcal{D})$, it can be answered in $\mathcal{D}$ in a "well-sorted way" too.

Theorem 4 (Completeness) *Let $\mathcal{D}$ be an order-sorted BAT in $\mathcal{L}^{OS}$, and $TR(\mathcal{D})$ be its corresponding Reiter's BAT. Then, for any query W in Reiter's situation calculus, W can be translated to a (well-sorted) query wrt $\mathcal{D}$, denoted as $os(W)$ below, s.t. $TR(\mathcal{D}) \models^{\text{ms}} tr(os(W)) \equiv W$. Furthermore, we have $TR(\mathcal{D}) \models^{\text{ms}} W$ iff $\mathbf{D} \models^{\text{os}}_{\mathcal{T}_{\mathcal{D}}} os(W)$.*

To prove Theorem 4, we first define some new concepts and prove a lemma.

Definition 5 Let $\mathcal{D}$ be a BAT in the order-sorted situation calculus $\mathcal{L}^{OS}$, and $TR(\mathcal{D})$ be its corresponding Reiter's BAT. Any term t in Reiter's situation calculus is a *possibly sortable term wrt $\mathcal{D}$*, if one of the following conditions holds:
(1) t is a variable of sort Act, $Object$ or Sit in $\mathcal{L}_{sc}$;
(2) t is a constant c, and $c : Q$ in $\mathcal{T}_{\mathcal{D}}$ (we say that the sort of c is Q wrt $\mathcal{D}$); or,
(3) t is of form $f(\vec{x}_{1..n})$, function declaration $f : \vec{Q}_{1..n} \rightarrow Q$ in $\mathcal{T}_{\mathcal{D}}$, for every i ($i = 1..n$), t_i either is a variable or is a non-variable term of sort Q'_i wrt $\mathcal{D}$ and $Q'_i \leq_{\mathcal{T}} Q_i$ in $\mathcal{T}_{\mathcal{D}}$ (we say that the sort of $f(\vec{t}_{1..n})$ is Q wrt $\mathcal{D}$).

Similarly, any atom $P(\vec{t}_{1..n})$ in Reiter's situation calculus (can be $t_1 = t_2$), which is well-sorted wrt $TR(\mathcal{D})$, is a *possibly*

sortable atom wrt $\mathcal{D}$, if for every i, t_i either is a variable or is a non-variable term s.t.:
(a) it is possibly sortable wrt $\mathcal{D}$; and
(b) $P\!:\!\vec{Q}_{1..n}$ is in $\mathcal{T}_{\mathcal{D}}$ (=: $\top\times\top$, respectively), the sort of t_i is Q'_i wrt $\mathcal{D}$ and $Q'_i \leq_{\mathcal{T}} Q_i$ wrt $\mathcal{D}$.

Given any $\mathcal{D}$ in order-sorted situation calculus, it is easy to see that every atom (term, respectively) in $TR(\mathcal{D})$ that can be considered as well-sorted wrt $\mathcal{D}$ is always a possibly sortable atom (term, respectively); while a possibly sortable atom (term, respectively) is not necessarily well-sorted wrt $\mathcal{D}$.

Lemma 3 *Let $\mathcal{D}$ be a BAT in the order-sorted situation calculus $\mathcal{L}^{OS}$, and $TR(\mathcal{D})$ be its corresponding Reiter's BAT. Then, for any atom $P(\vec{t}_{1..n})$ (can be $t_1 = t_2$) that is well-sorted in $\mathcal{L}_{sc}$ but not possibly sortable wrt $\mathcal{D}$, we have $TR(\mathcal{D}) \models^{\mathrm{ms}} P(\vec{t}_{1..n}) \equiv false$.*

Now we define a function which transforms a formula in $\mathcal{L}_{sc}$ wrt $TR(\mathcal{D})$ to a well-sorted formula in $\mathcal{L}^{OS}$ wrt $\mathcal{D}$.

Definition 6 Let $\mathcal{D}$ be a BAT in the order-sorted situation calculus $\mathcal{L}^{OS}$, $TR(\mathcal{D})$ be its corresponding Reiter's BAT and W be a regressable sentence in $\mathcal{L}_{sc}$ wrt the background BAT $TR(\mathcal{D})$. Then, function $os(W)$ is defined recursively as follows.

1. If W is either of the form $(\forall x)W_1$, $(\exists x)W_1$, where the default sort of x is Q (either $Object$, Act or Sit) in $TR(\mathcal{D})$, then $os((\forall x)W_1) \overset{def}{=} (\forall x\!:\!Q)os(W_1)$, and $os(\exists x.W_1) \overset{def}{=} (\exists x : Q)os(W_1)$.
2. If W is one of the form $\neg W_1$, $W_1 \wedge W_2$, $W_1 \vee W_2$, then $os(\neg W_1) \overset{def}{=} \neg os(W_1)$, $os(W_1 \wedge W_2) \overset{def}{=} os(W_1) \wedge os(W_2)$, $os(W_1 \vee W_2) \overset{def}{=} os(W_1) \vee os(W_2)$.
3. If W is atomic and not possibly sortable, then $W \overset{def}{=} false$.
4. If W is atomic and possibly sortable, assume that $var(W) = \langle x_1, \cdots, x_n\rangle$ is the vector of free variables appeared from left to right in W (including repeated ones). For each $i = 1..n$, suppose that x_i appears as an argument of a function f_i in some term or as an argument of a predicate P_i in W. Let Q_i be the sort appeared in the k_i-th position of the declaration of f_i (P_i, respectively), if x_i appears in the k_i-th position of f_i (P_i, respectively) in W. Then, let $I_W = \{i \mid x_i \in var(W), Q_i \leq_{\mathcal{T}} Object, Q_i \neq Object\}$, and $\vec{y}\!:\!\vec{Q} = \{y_i\!:\!Q_i \mid i \in I_W\}$, where y_i's are auxiliary variables never appeared in W and each y_i is distinct from others. And, $os(W) \overset{def}{=} (\exists\vec{y} : \vec{Q})(W_0 \wedge \bigwedge_{i\in I_W} x_i = y_i)$, where W_0 is obtained from substituting each x_i with y_i for $i \in I_W$.

Proof sketch for Theorem 4. First, for any query W in Reiter's situation calculus, let $W' = os(W)$. By using structural induction and Lemma 3, it is easy to prove that W' is a well-sorted query wrt $\mathcal{D}$ in OSL and $TR(\mathcal{D}) \models^{\mathrm{ms}} W \equiv tr(W')$. Then, by Theorem 3 and $TR(\mathcal{D}) \models^{\mathrm{ms}} W \equiv tr(W')$, it is easy to see that $\mathbf{D} \models^{\mathrm{os}}_{\mathcal{T}_{\mathcal{D}}} W'$ iff $TR(\mathcal{D}) \models^{\mathrm{ms}} tr(W')$ iff $TR(\mathcal{D}) \models^{\mathrm{ms}} W$. Proof details are omitted due to the space limitations. But, we provide some examples below to illustrate the statement.

Example 5 Here are simple examples of computing $os(W)$ from W in $\mathcal{L}_{sc}$. Consider the $TR(\mathcal{D})$ in Example 4. Let $On(Boston, T_1, S_1)$ (denoted as W_3) be a query in $\mathcal{L}_{sc}$, where S_1 is some situation instance. According to the way $TR(\mathcal{D})$ is constructed, we have $TR(\mathcal{D}) \models^{\mathrm{ms}} On(o, t, s) \supset Box(o)$ and $TR(\mathcal{D}) \models^{\mathrm{ms}} \neg Box(Boston)$. So, $TR(\mathcal{D}) \models^{\mathrm{ms}} W_3 \equiv false$. Hence, $os(W_3) \overset{def}{=} false$.
Let W_4 be $\forall s.\exists o.\neg InCity(o, Pasadena, s)$, which is also a query in $\mathcal{L}_{sc}$, where $o : Object$ and $s : Sit$ hold by default. Then, $os(W_4)$ is $\forall s : Sit.\exists o : Object.\neg(\exists b : MovObj.b = o \wedge InCity(b, Pasadena, s))$, since $TR(\mathcal{D}) \models^{\mathrm{ms}} InCity(o, c, s) \supset MovObj(o) \wedge City(c)$. And it is easy to prove that $TR(\mathcal{D}) \models^{\mathrm{ms}} W_4 \equiv tr(os(W_4))$.

Computational Advantages of $\mathcal{L}^{OS}$

In this section, we discuss the advantages of using OSL and the order-sorted regression operator based on it.

Given any BAT $\mathcal{D}$ in $\mathcal{L}^{OS}$, it is easy to see that Reiter's regression operator $\mathcal{R}$ (Reiter 2001) still can be applied to (well-sorted) regressable formulas (wrt $\mathcal{D}$). Moreover, one can prove that $\mathcal{R}[W]$ is a formula in $\mathcal{L}^{OS}$ uniform in S_0 and $\mathbf{D} \models^{\mathrm{os}}_{\mathcal{T}_{\mathcal{D}}} W \equiv \mathcal{R}[W]$. However, using the order-sorted regression operator $\mathcal{R}^{os}$ sometimes can give us computational advantages in comparison to using Reiter's regression operator $\mathcal{R}$. But first of all, we show that the computational complexity of using $\mathcal{R}^{os}$ is no worse than that of $\mathcal{R}$.

For the regression operator $\mathcal{R}$ that can be used either in $\mathcal{L}^{OS}$ or in $\mathcal{L}_{sc}$ ($\mathcal{R}^{os}$ used in $\mathcal{L}^{OS}$, respectively), we can construct a *regression tree* rooted at W for any regressable query W in either language. Each node in a regression tree of $\mathcal{R}[W]$ ($\mathcal{R}^{os}[W]$, respectively) corresponds to a sub-formula computed by regression, and each edge corresponds to one step of regression according to the definition of the regression operator. In the worst case scenario, for any query W in $\mathcal{L}^{OS}$, the regression tree of $\mathcal{R}^{os}[W]$ will have the same number of nodes as the regression tree of $\mathcal{R}[W]$ (and linear to the number of nodes in the regression tree of $\mathcal{R}[tr(W)]$ wrt $TR(\mathcal{D})$). Moreover, based on the assumption that our sort theory of $\mathcal{D}$ is simple with empty equational theory, whose corresponding sort hierarchy is a meet semi-lattice, finding a unique (well-sorted) MGU takes the same time as in the unsorted case (Schmidt-Schauβ 1989; Jouannaud & Kirchner 1991; Weidenbach 1996). Hence, the overall computational complexity of building the regression tree of $\mathcal{R}^{os}[W]$ is at most linear to the size of Reiter's regression tree.

Theorem 5 *Consider any regressable sentence W with a background BAT $\mathcal{D}$ in order-sorted situation calculus $\mathcal{L}^{OS}$. Then, in the worst case scenario, the complexity of computing $\mathcal{R}^{os}[W]$ is the same as that of computing $\mathcal{R}[W]$, which is also the same as the complexity of computing $\mathcal{R}[tr(W)]$ in the corresponding Reiter's BAT $TR(\mathcal{D})$.*

On the other hand, under some circumstances, the regression of a query in $\mathcal{L}^{OS}$ using $\mathcal{R}^{os}$ instead of $\mathcal{R}$ will give us computational advantages. Consider any query (i.e., a regressable sentence) W with a background BAT $\mathcal{D}$ in $\mathcal{L}^{OS}$. Then, the computation of $\mathcal{R}^{os}[W]$ wrt $\mathcal{D}$ can sometimes terminate earlier than that of $\mathcal{R}[W]$ wrt $\mathcal{D}$, and also earlier than the computation of $\mathcal{R}[tr(W)]$ wrt $TR(\mathcal{D})$. In particular, we have the following property.

Theorem 6 *Let a regressable formula W have the syntactic form $t_{1,1} = t_{1,2} \wedge \ldots \wedge t_{m,1} = t_{m,2} \wedge W_1$, with any background order-sorted BAT $\mathcal{D}$ in $\mathcal{L}^{OS}$. Let the size of W (including the length of the terms in W) be n. If there is no well-sorted mgu for equalities between terms, then Computing $\mathcal{R}^{os}[W]$ runs in time $O(n)$, while computing $\mathcal{R}[W]$ wrt $\mathcal{D}$ ($\mathcal{R}[tr(W)]$ wrt $TR(\mathcal{D})$) runs in time $O(2^n)$. Moreover, the size of the resulting formula of $\mathcal{R}^{os}[W]$, which is $false$, is always constant, while the size of the resulting formula using $\mathcal{R}$ is in $O(2^n)$.*

According to the definition of Reiter's regression operator, the equalities will be kept and regression will be further performed on W_1 (or on $tr(W_1)$ in $TR(\mathcal{D})$, respectively), which in general takes exponential time wrt the length of W_1 and causes exponential blow-up in the size of the formula. Once Reiter's regression has terminated, a theorem prover will find that the resulting formula is false either because there is no mgu for terms when reasoning is performed in $\mathcal{L}^{OS}$ (or, due to the clash between sort related predicates when reasoning in $\mathcal{L}_{sc}$, respectively). Hence, using the order-sorted regression operator can sometimes prune brunches of the regression tree built by $\mathcal{R}$ exponentially (wrt the size of the regressed formula), and therefore save computation time significantly.

Example 6 Consider the BAT $\mathcal{D}$ from Example 1. Let W_5 be a $\mathcal{L}^{OS}$ query (i.e., a (well-sorted) regressable sentence) $InCity(T_1, Pasadena, do(drive(T_1, Boston, Pasadena), S_1))$, where S_1 is a well-sorted ground situation term that involves a long sequence of actions. According to the SSA of $InCity$, at the branch of computing $\mathcal{R}^{os}[\exists b : Box.b = T_1 \wedge On(b, t, S_1)]$ in the regression tree, since there is no well-sorted mgu for (b, T_1), the application of order-sorted regression equals to $false$ immediately. However, using Reiter's regression operator (no matter in $\mathcal{D}$ or in $TR(\mathcal{D})$), his operator will keep doing useless regression on $On(b, t, S_1)$ until getting (a potentially huge) sub-formula uniform in S_0. Once his regression has terminated, such sub-formula will also be proved equivalent to $false$ wrt the initial theory ($\mathcal{D}_{S_0}$ or $TR(\mathcal{D})_{S_0}$, respectively) using a theorem prover, for the same reason as above.

In addition, since our sort theory of a BAT $\mathcal{D}$ in $\mathcal{L}^{OS}$ is finite and it has one and only one declaration for each function and predicate symbol, for any query W (wrt $TR(\mathcal{D})$) in $\mathcal{L}_{sc}$, it takes linear time (wrt the length of the query) to find a well-sorted formula $os(W)$ in $\mathcal{L}^{OS}$ that satisfies Theorem 4. But, reasoning whether $\mathbf{D} \models^{\mathrm{os}}_{\mathcal{T}_\mathcal{D}} os(W)$ (starting from finding $os(W)$) sometimes can terminate earlier than finding whether $TR(\mathcal{D}) \models^{\mathrm{ms}} W$. In particular, we have

Theorem 7 *Assume that $W = F(\vec{t}, do([\alpha_1, \cdots, \alpha_n], S_0))$ is an atomic fluent instance in $\mathcal{L}_{sc}$ that includes an ill-sorted ground term wrt $\mathcal{D}$ (e.g., W_3 in Example 5). Then, it takes at most linear time to terminate reasoning by computing the corresponding $os(W)$ (which is $false$).*

Observe that reasoning about $TR(\mathcal{D}) \models^{\mathrm{ms}} W$ directly, for the formula W mentioned in Theorem 7, using regression $\mathcal{R}$ could result in a exponentially large regression tree when computing $\mathcal{R}[W]$. Also, the size of the resulting formula can be exponentially larger than that of W. Moreover, it still needs further computational steps to find whether $TR(\mathcal{D})_{S_0} \cup TR(\mathcal{D})_{una} \models^{\mathrm{ms}} \mathcal{R}[W]$.

Conclusions

We propose a logical theory for reasoning about actions wrt a taxonomy of objects based on OSL. We also define a regression-based reasoning mechanism that takes advantages of sort theories, and discuss the computational advantages of our theory. One possible future work can be extending our logic to hybrid order-sorted logic (Cohn 1989; Bierle *et al.* 1992; Weidenbach 1996). Another possibility is to consider efficient reasoning in our framework by identifying specialized classes of queries or decidable fragments (Abadi, Rabinovich, & Sagiv 2007). Finally, we are planning to work on an efficient implementation of our theory.

References

Abadi, A.; Rabinovich, A. M.; and Sagiv, M. 2007. Decidable fragments of many-sorted logic. In *LPAR*, volume 4790 of *Lecture Notes in Computer Science*, 17–31. Springer.

Bierle, C.; Hedtstück, U.; Pletat, U.; Schmitt, P. H.; and Siekmann, J. 1992. An order-sorted logic for knowledge representation systems. *Artificial Intelligence* 55(2-3):149–191.

Classen, J.; Eyerich, P.; Lakemeyer, G.; and Nebel, B. 2007. Towards an integration of golog and planning. In *20th International Joint Conference on Artificial Intelligence (IJCAI-07)*. AAAI Press.

Cohn, A. G. 1987. A more expressive formulation of many sorted logic. *J. Autom. Reason.* 3(2):113–200.

Cohn, A. G. 1989. Taxonomic reasoning with many sorted logics. *Artificial Intelligence Review* 3(2-3):89–128.

Ghallab, M.; a. Howe; Knoblock, C.; McDermott, D.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. PDDL the planning domain definition language. Technical report, Yale Center for Computational Vision and Control, Technical Report CVC TR-98-003/DCS TR-1165.

Goguen, J. A., and Meseguer, J. 1987. Remarks on remarks on many-sorted equational logic. *SIGPLAN Notices* 22(4):41–48.

Hayes, P. J. 1971. A logic of actions. *Machine Intelligence* 6:495–520.

Herbrand, J. 1971. *Logical Writings*. Cambridge: Harvard University Press. Warren D. Goldfarb (ed.).

Jouannaud, J.-P., and Kirchner, C. 1991. Solving equations in abstract algebras: A rule-based survey of unification. In *Computational Logic - Essays in Honor of Alan Robinson*, 257–321. MIT Press.

Oberschelp, A. 1962. Untersuchungen zur mehrsortigen quantorenlogik (in German). *Mathematische Annalen* (145):297–333.

Oberschelp, A. 1990. Order sorted predicate logic. In *Sorts and Types in Artificial Intelligence*, volume 418 of *Lecture Notes in Computer Science*, 8–17. Springer.

Reiter, R. 2001. *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems*. MIT Press.

Schmidt-Schauβ, M. 1989. *Computational aspects of an order-sorted logic with term declarations*. New York: Springer-Verlag.

Schmidt, A. 1938. Über deduktive theorien mit mehreren soften von grunddingen. *Mathematische Annalen* (115):485–506.

Walther, C. 1987. *A many-sorted calculus based on resolution and paramodulation*. San Francisco: Morgan Kaufmann.

Wang, H. 1952. Logic of many sorted theories. *Symbolic Logic* 17(2):105–116.

Weidenbach, C. 1996. Unification in sort theories and its applications. *Annals of Math. and AI* 18(2/4):261–293.

Autonomous Learning of Commonsense Simulations

Benjamin Johnston and Mary-Anne Williams
University of Technology, Sydney
Ultimo, Sydney, New South Wales, Australia
johnston@it.uts.edu.au

Abstract

Parameter-driven simulations are an effective and efficient method for reasoning about a wide range of commonsense scenarios that can complement the use of logical formalizations. The advantage of simulation is its simplified knowledge elicitation process: rather than building complex logical formulae, simulations are constructed by simply selecting numerical values and graphical structures. In this paper, we propose the application of machine learning techniques to allow an embodied autonomous agent to automatically construct appropriate simulations from its real-world experience. The automation of learning can dramatically reduce the cost of knowledge elicitation, and therefore result in models of commonsense with breadth and depth not possible with traditional engineering of logical formalizations.

Introduction

Comirit is an open-ended hybrid architecture for commonsense reasoning. We have previously described (Johnston and Williams 2008) how Comirit is a generalization of the method of analytic tableaux; combining rich 3D simulation with formal logic. In this paper we extend the architecture so that it supports *autonomous learning* in addition to deduction. We demonstrate the system by implementing stochastic search and an auto-associative network in the framework: this enables our experimental system to autonomously acquire and maintain reliable knowledge of novel objects and their behaviors.

At Commonsense 2007 (Johnston and Williams 2007), we argued that simulations are a potentially rich resource of commonsense knowledge and are an expressive and efficient mechanism for commonsense reasoning. The potential for breadth and depth is clearly evident when one considers the advances in modern animations and computer games: modern games offer realistic open-ended ‘sandbox’ environments for unlimited experimentation and interaction. We showed that a generic graph-based representation can be used to rapidly create similarly rich and realistic simulations with minimal software development, and thereby produce the possibility of extracting and directly reasoning with the knowledge that would otherwise be only implicitly represented in simulation. Generic graph-based representations further simplify the knowledge elicitation process to a task that can be performed as a routine software development process without the expense (or concern for a shortage) of skilled logicians or philosophers.

While simulation may be seen as a powerful heuristic for deducing possible and likely future states from current conditions, a weakness of simulation is that it must follow the ‘arrow-of-time’. That is, it is impossible to simulate a complex situation in *reverse* to deduce likely causes or precursors of a situation: one cannot simulate spilled milk in reverse to discover a likely cause—it is far easier to simulate the outcome from a particular cause such as dropping an open milk carton onto the floor. When greater deductive power is required than that of forward-running simulations, it is therefore necessary to augment simulation with other mechanisms.

We proposed (Johnston and Williams 2008) the integration of simulation and logical deduction as a way of combining the efficiency and richness of simulation with the power of logic. Our framework generalized the method of analytic tableaux to allow both logical terms and simulation objects within a single search structure. A single unifying principle based on the idea of searching through spaces of possible worlds enabled these disparate mechanisms to be harmoniously combined in a single system.

While our framework offers a mechanism for commonsense reasoning, we acknowledge that an effective system includes not only a reasoner, but also a comprehensive commonsense knowledge-base. The 20-year Cyc project (Panton *et al.* 2006) to create a commonsense knowledge-base serves as a clear demonstration that such engineering can be exorbitantly expensive. We therefore wondered, “is it possible to automate knowledge acquisition?” Our graph-based simulations simplify the engineering process, so we initially expected that powerful semi-automatic engineering tools could permit rapid knowledge elicitation. In designing these tools, it soon became apparent that the graph representation could allow *fully autonomous* learning and generalization of aspects of the behavior of novel objects.

The purpose of this paper is therefore to explain how learning may be incorporated into our commonsense reasoning framework. We view cumulative learning as an iterative process of hypotheses generation and selection: this perspective can be elegantly incorporated into a tableaux reasoning framework by ranking branches of the search tree and allowing for factoring of sub-problems. The purpose of these modifications is to allow branches of a tableau to contain hypotheses, and for the search algorithm to focus only on those branches with the best hypothesis.

This paper begins with a brief overview of the existing framework: the underlying representation used by simulation, and the hybrid reasoning strategy. We then explain how the framework is extended to allow learning, and conclude with a concrete exploration into how stochastic hill-climbing and auto-associative networks may be incorporated into this framework to allow a system to autonomously learn simulations of novel objects.

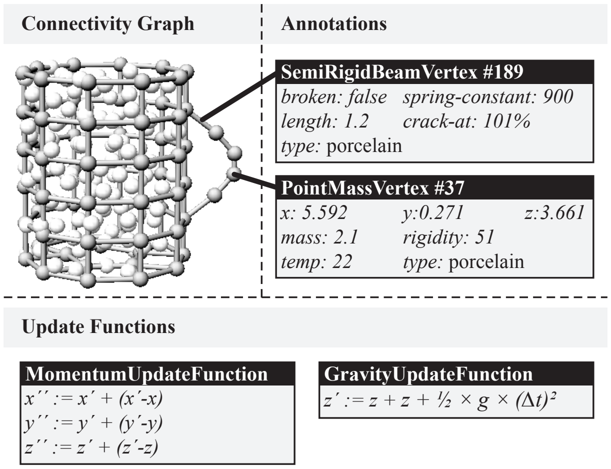

Figure 1: *Examples of the components of a simulation*

Simulation

In the Comirit framework, simulations are used as the underlying mechanism and representation for large scale commonsense knowledge. Not all knowledge can be represented efficiently in simulations (*e.g.*, 'What is the name of the Queen of England?'), but simulation works well in problems governed by simple laws (such as physics) and so simulation is used in the framework wherever possible.

Comirit simulations are a sophisticated generalization of an early proposal by Gardin and Meltzer (1989); extended to support 3D environments and non-physical domains. Comirit simulations are constructed from a graph-based representation. The fundamental structure of a problem is first approximated by a graph. The graph is then annotated with frame-like structures, and simulation proceeds by the iterative update of the annotations by update functions.

The formal details of simulation are not needed to understand this paper, so we will use illustrative examples. Readers interested in the formal details should refer to our previous publication (Johnston and Williams 2007).

A Comirit simulation consists of the following parts:

1. a system clock that increments by finite intervals
2. a (relatively) static graph representation that models the underlying structure of a problem domain,
3. a set of highly dynamic annotations that record the state of a simulation, and
4. a static set of computable functions or constraints that update the annotations with each iteration of the system clock and thereby drive the computation of the simulation.

This representation is intentionally generic. We claim that it can be used to represent simulations from any rule-driven problem domain including physical, social, legal, economic and purely abstract realms. Our research has emphasized physical reasoning and naïve physics, so we will illustrate simulation and learning through examples based upon 3D simulations of physical models.

Consider a simple domestic robot facing a physical reasoning problem: *given a mug filled with coffee, is it 'safe' to perform fast movements to carry the mug?* In the Comirit framework, the robot considers the problem by internal simulations of the scenario; testing whether a simulated mug is damaged by fast movement, or if such motion causes damage to the environment by spilling coffee.

The generic graph structure is used to represent the underlying structure of the problem. For example, the mug of coffee can be approximated as a mesh of point masses connected by semi-rigid beams. A visualization of such a graph appears in Figure 1. Note that both the spheres and beams are vertices of the underlying graph, their connectivity is recorded by edges in the graph.

Each vertex of the simulation graph (*i.e.*, each point mass *and* each semi-rigid connecting bar) is annotated with a set of frame-like attributes such as the current 3D position, local mass distribution, rigidity, physical state, temperature and whether the local structure has been broken (due to over-stressing). Examples of particular annotations and values also appear in Figure 1. Note that these annotations represent only *local* properties of the simulation. The total mass of the mug of coffee is equal to the *sum* of all of the `mass` attributes.

Simulation proceeds by the iterative update of annotation values. Newton's laws of motion are applied to each of the point masses, and Hooke's law (describing the behavior of a spring) is applied to the connecting beams. Figure 1 illustrates update functions for the laws of momentum and gravity. Note that these functions have only short-term and local effects.

The combined effect of iteratively computing local updates on the annotations is emergent behavior that closely resembles the actual behavior of real world scenarios. Laws of physics are simple at the microscopic scale. The macroscopic shape of an object, its centre of mass, its rotational inertia and its viscosity or brittleness vastly complicate the physical laws of motion of large bodies, but these simply emerge from iteration of simple laws at the microscopic scale. Indeed, this method of simulation may be seen as a variation on the Euler method of numerical integration. Simulation effectively performs numerical integration over the fundamental laws of physics that are expressed as stepwise differential equations in the update functions.

If we run a mug of coffee simulation we may result in an outcome such as that depicted in Figure 2. Symbolic results are reported by simple routines that inspect the state of the simulation to determine if, for example, any semi-rigid bars have broken (in the case of a '**`broken`**' symbol), or if any liquid is no longer contained by the mug (in the case of a '**`mess`**' symbol). This outcome would imply that the robot should not use fast movements with the mug.

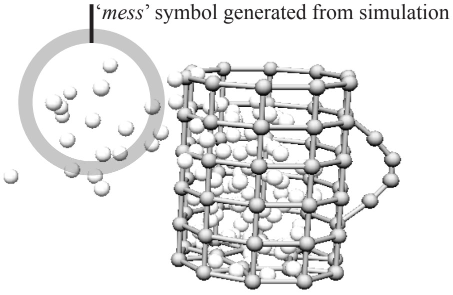

Figure 2: *Simulating a 'fast move' on a mug of coffee*

Hybrid Architecture

Simulation is a powerful and efficient tool for commonsense reasoning, but it only supports a ‘forward chaining’ inference mode: it is therefore an incomplete solution for general purpose commonsense reasoning. We integrate simulation with logical deduction in a hybrid architecture in order to combine the strengths and complement the weaknesses of each mechanism. That is, we use the deductive power of a general-purpose logical reasoner to make up for the inflexibility of simulation.

In combining simulation and logic, blackboard architectures, tuple spaces and agent architectures serve as obvious choices for implementation: they have a long history of application to problems of integration in intelligent systems. Unfortunately, our experience is that the conceptual mismatch between simulation and logic is such that application of these integration techniques eventually results in systems that are unworkably complex and difficult to maintain. Instead, a clear and unifying abstraction is required to harmonize the semantics of the reasoning mechanisms. We claim that this can be achieved by our hybrid architecture that performs logical deduction with the method of analytic tableaux, and interprets both simulation and logical deduction as operations over spaces of worlds.

The method of analytic tableaux (Hähnle 2001) is an efficient method of mechanizing logical theorem proving. Analytic tableaux have been successfully applied to large problems on the semantic web, and there is a vast body of literature on their efficient implementation (*ibid.*). The method constructs trees (tableaux) through the syntactic decomposition of logical expressions, and then eliminates branches of the tree that contain contradictions among the decomposed atomic formulae. Each branch of the resultant tableau may be seen as a partial, disjunction-free description of a model for the input formulae. The crucial insight is that if a tableau algorithm is given knowledge of the world and a query as logical input, then the conjuction of the atomic formulae in a branch of the resultant tree represents a *space of worlds that satisfy the query*.

The tableau method *and* simulation can thereby be unified through this common abstraction. The tableau algorithm generates spaces of worlds, and simulation expands upon knowledge of spaces of worlds (*i.e.*, forward chains to future states based on current states). In our framework, we perform commonsense reasoning by generalizing the tableau so that it can contain non-logical terms such as simulations, functions and data-structures in addition to the standard logical terms of traditional tableaux. Deduction proceeds by application of both tableau rules that expand, fork or close branches of the tree, and simulations that can expand and close branches of the tree.

The full details of this method appear in our earlier publications, but we will review the principles here by way of a simplified example. Consider the following scenario:

> *A household robot needs to move an object across a table. Its actuators can perform a soft or a hard movement. It is unsafe to move any object ‘quickly’. What commands may be sent to the actuators?*

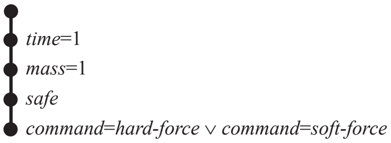

Figure 3a: *First, each conjunct in the original query is expanded into separate nodes.*

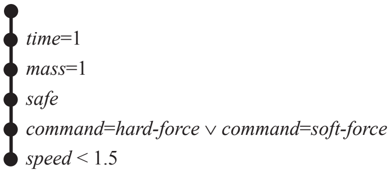

Figure 3b: *The term ‘safe’ is expanded per its definition.*

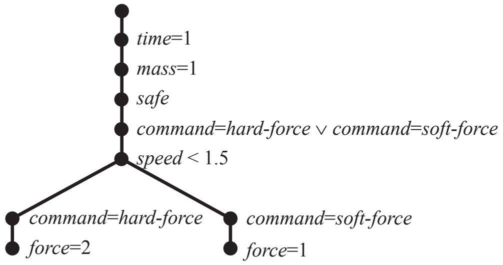

Figure 3c: *The tableau is forked into two branches: one branch for each disjunct in the fifth node. ‘Hard-force’ and ‘soft-force’ are then expanded per their definitions. Logical deduction has now ‘stalled’: no more logical rules apply.*

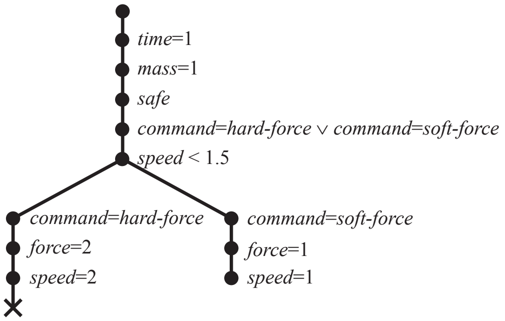

Figure 3d: *Simulation is now invoked. As a result, the left branch becomes inconsistent (speed=2 and speed<1.5). The right branch remains open and therefore describes a scenario satisfying the original query (i.e., the robot can safely use soft-force).*

For the convenience of our example calculations, let us assume that the mass of the object is 1kg, the soft force is 1N, the hard force is 2N, the object is simply pushed for 1s and unsafe speeds are 1.5ms^{-1} or higher. Furthermore, we assume the following highly simplified and abstracted simulation[1]:

```
function simulate(Mass, Force, Time):
    set Speed := Time * Force / Mass
    return {speed = Speed}
```

We can then convert the scenario to a logical form:

$$\mathit{time}{=}1 \wedge \mathit{mass}{=}1 \wedge \mathit{safe} \wedge (\mathit{command{=}hard\text{-}force} \vee \mathit{command{=}soft\text{-}force})$$

With this logical form, we may then apply the method of analytic tableau and simulation to find models that satisfy

[1] Note that while highly simplified, this algorithm has similar constraints to real simulations: the inputs are numerical and fully specified, and the algorithm can only be used in the ‘forward’ direction.

the formula. The algorithm proceeds in the steps illustrated in Figures 3a–3d.

When the algorithm terminates there is a tree with only one open branch. Reading atomic formulae along that remaining branch, we see that it describes a world in which the *command=soft-force* action is applied and the object moves safely at 1ms^{-1}.

Thus, we have used both simulation and logical deduction in a single mechanism to solve a (simplified) commonsense reasoning problem. Details such as data structures, methods for prioritizing computation, search strategies and output variables have been omitted from this example for clarity, however these can be found in our earlier publication (Johnston and Williams 2008).

Integrating Learning

Commonsense reasoning requires more than a hybrid method of deduction: it depends on the availability of rich and accurate knowledge of the world. In contrast to logical methods that depend on highly skilled logicians painstakingly encoding their intuitions into formal axiomatizations, a small number of fundamental laws are first implemented in a simulation, and then descriptions of the world can be readily added to a simulation in a simple two stage process:

1. Structuring the underlying simulation graph to match the observed structure of the situation to be simulated (such as creating a 3D wire-frame model),
2. Configuring the annotations on the graph so that the simulations closely predict reality.

In our own experiences with constructing simulations, we observed that these tasks involved little mental effort but were a tedious process of careful tuning of parameters to match reality (*e.g.*, trial-and-error to determine an appropriate spring constant to simulate a rubber ball). We begun creating tools to support this process, but quickly realized that tedious and undemanding tasks are ideal candidates for full automation. Consider the following:

1. The update functions in a simulation are static: they are easily implemented manually, and are rarely changed. For example, once the laws of Newtonian physics have been implemented, they can be used in simulations of almost all mechanical systems. While we currently view this as outside the realm of feasible automation, we do not consider this effort to be substantial.
2. The underlying static graph can often be directly observed from the environment. In the case of physical simulations, a wire-frame approximation can be automatically assembled from the 3D volumes reconstructed from moving camera images, stereoscopic vision, LASER sensing, 3D scanning, LIDAR, Z-cameras, direct physical contact or other scanning/imaging techniques.
3. The annotations that guide the dynamic behaviour of simulations are typically numeric so their values may be learnt with standard machine learning techniques (*i.e.*, by search for parameters that minimize the error between simulation and observed reality).

For example, if we have a household robot that uses simulation as an underlying representation, then the robot can acquire knowledge of a novel object (say a mug of coffee) by building a wireframe model from a robust 3D shape reconstruction algorithm, and then searching for annotations that match the observed behaviour of the object. Observations can be used to test hypotheses about annotation values by simulating the behaviour of the novel object and comparing them to the observation. In our framework, the robot automatically and continuously learns object and annotations in a background 'process' by constantly simulating from recent observations, and testing that expectations match the current observation. This is illustrated in Figure 4.

The relationship between input and output in a simulation is difficult to compute analytically (indeed, if there were simple analytical solutions, it is unlikely that simulation would be applied to the problem in the first place). Annotations must be computed by numerical optimization or machine learning algorithms. In particular, we intend to use a greedy search to find these values (any other generate-and-test algorithm can be used, as appropriate to the problem: genetic programming, beam search or simulated annealing).

How then, can optimization be incorporated into our unifying abstraction of search over spaces of possible worlds? Optimization requires comparison of separate spaces of worlds, and therefore involves comparing separate parts of the search space or separate branches of a tableau. Unfortunately, the standard tableau algorithm only permits manipulation or inspection of a single branch. We avoid this problem by introducing an (incomplete) ordering over branches, and then modifying the tableau algorithm so that it searches for minimal models. A branch in a tableau is no longer considered 'open' simply if it is consistent—it is open if it is either *consistent and unordered*, or else *it has an ordering and is minimal among all other consistent and ordered branches*. That is, the algorithm considers all unordered branches, and only one ordered branch. (Note also that if the minimal ordered branch is found inconsistent, the next most minimal branch is then considered again.)

We *could* define the ordering outside of the tableau. For example: "Branch *a* is smaller than branch *b* if the error of the first hypothesis in *a* is smaller than that of the first hypothesis in *b*. If the first hypothesis in *a* and *b* are equal, then the ordering is determined by the second hypothesis (and so on)". However, such ordering rules are inconvenient because they are defined outside the tableau.

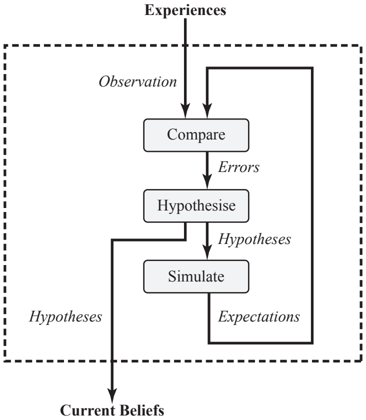

Figure 4: *Background learning process*

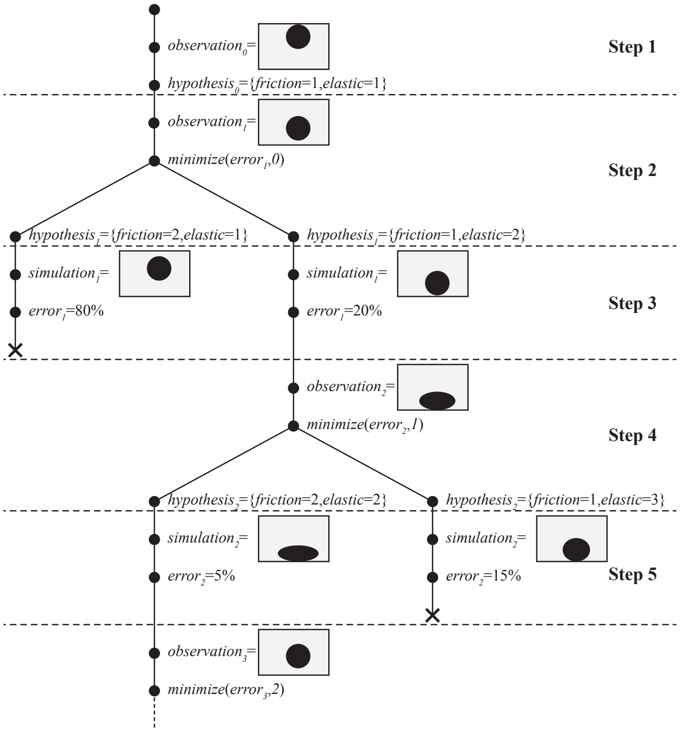
observation$_0$=
hypothesis$_0$={*friction*=1,*elastic*=1}
Step 1
observation$_1$=
minimize(*error*$_1$,0)
Step 2
hypothesis$_1$={*friction*=2,*elastic*=1}
hypothesis$_1$={*friction*=1,*elastic*=2}
simulation$_1$=
error$_1$=80%
simulation$_1$=
error$_1$=20%
Step 3
observation$_2$=
minimize(*error*$_2$,1)
Step 4
hypothesis$_2$={*friction*=2,*elastic*=2}
hypothesis$_2$={*friction*=1,*elastic*=3}
simulation$_2$=
error$_2$=5%
simulation$_2$=
error$_2$=15%
Step 5
observation$_3$=
minimize(*error*$_3$,2)

Figure 5: *An example of learning in a tableau*

Instead, we define an ordering with symbols stored within the tableau. We hold the set of propositions **minimize**(*VariableName*, *Priority*) as tautologically true in the standard tableau algorithm, but use them in evaluating the order of the branch. The *Priority* is a value from a totally-ordered set (such as the integers) and indicates the order in which the values of variables are sorted: branches are first sorted by the highest priority variable, then equal values are sorted using the next highest priority variable, and so on.

Consider our household robot example again. If the robot encounters a novel object it will need to find suitable parameters to simulate and therefore reason about the object. If there is another agent or a designated 'teacher' demonstrating how to handle the object, it has a ready stream of observations for learning. If the object is simply sitting alone, it may need to apply its most conservative and gentle action to the object to gather some initial data. The robot can then use a stochastic hill-climbing strategy on its observations to learn about the object:

1. Initially, a default hypothesis about the values of annotations is assigned to the novel object.
2. When a new observation arrives, the robot generates a set of alternate hypotheses (random perturbations, in the case of a stochastic hill-climbing strategy) as a disjunction in the tableau: this disjunction produces new branches in the tableau.
3. The alternate hypotheses are each simulated from the prior observation in order to generate predictions for the current observation. The error between expectation and reality is computed.
4. The best hypothesis is implicitly chosen by the tableau algorithm, due to its preference for minimally ordered branches. The algorithm continues again with step 2.

Note also that because our tableaux can contain logic, simulations and functions, the system may use logical constraints or ad-hoc 'helper functions' even when searching for values in a simulation (*e.g.*, a constraint such as *mass* > 0, or a custom hypothesis generator that samples the problem space in order to produce better hypotheses).

An example of learning the behaviour of a falling ball by hypothesis search in a tableau appears in Figure 5:

Step 1: The tableau initially contains the first observation of a ball and the initial hypothesis generated (many other control objects, meshes, functions and other data will be in the tableau, but these are not shown for simplicity).

Step 2: The system observes movement in the ball. It generates new hypotheses, seeking to find a hypothesis with minimal error.

Step 3: The system simulates from *hypothesis*$_0$. The result of the simulation is compared with *observation*$_1$ to determine the error in the hypothesis. The right branch has smaller error so the left branch is no longer open.

Step 4: As with Step 2, the system observes more movement and generates new hypotheses, further refining the current hypothesis.

Step 5: The system then simulates as with Step 3, but this time the left branch is minimal. In the following steps, the algorithm continues yet again with more new observations and further hypothesizing…

Experimental Results

We tested an implementation of this technique in a simple virtual environment: boxes and balls of varying sizes, masses and elasticity (some were rigid, others quite elastic), were pushed by variable forces for variable periods of time[2]. A virtual 2D 'camera' observed the interactions, and the Comirit learning algorithm was used to construct models from observations, generate hypotheses and simulations, and compare observation with simulations. The accuracy (as tested across many experiments) was surprisingly high:

1. A single observation pair is sufficient to learn annotations for simulating with 94% pixel-by-pixel accuracy.
2. Four observation pairs bring this accuracy up to 97.5% accuracy.
3. Further observations result in small, incremental improvements, approximately halving the error with each doubling of the number of new observations.

In our subsequent experiment, we used a broader concept of 'hypothesis'. Rather than learning the isolated annotations of individual objects, the hypothesis space was a self-organizing map (SOM) with feature vectors that include observable appearance and hidden parameters. The system retrieves an initial hypothesis for the annotations of a new object by searching the self-organizing map with a partial vector describing only the object's observable appearance. When the system observes errors between observation and simulated expectation, a new and complete feature vector is

[2] *i.e.*, Two observable object parameters (*shape*, *size*), two hidden parameters to be learnt (*mass*, *elasticity*) and two known observation parameters (*force*, *duration*)

generated and this is updated into the self-organising map.

We hoped to demonstrate an ability to generalize knowledge, so we added structure to our learning problem:

1. Color was inversely correlated with elasticity (we observe a similar effect in real life when shiny metallic objects in the real world are usually rigid).
2. Heavy (dense) objects were generally inelastic (as we often observe in real life).
3. Round objects were generally inelastic (we also observe similar correlations between shape and behavior in real life: a mug is generally rigid so that it may safely hold hot liquids).

To our surprise, not only did the system build a self-organizing map that captured the problem structure (and therefore allowed it to correctly generalize its learning about previously unseen 'heavy boxes'), but the self-organizing map improved the speed at which the algorithm learnt. This improvement is due to the map offering better hypotheses during early learning by generalizing from similar cases. The ability of the framework to rapidly learn from very few observations, meanwhile prevented it from being committed to its SOM hypothesis when it encountered 'outlier' objects.

In this latter trial we used a less aggressive hill-climbing strategy to allow greater exploration of the search space and better test the advantage of a SOM. With this weaker strategy, learning on a single object requires up to 8 observations to achieve 90% accuracy. A randomly initialized SOM has less than 5% accuracy, but after observing 14 objects once each, it achieved 60% accuracy, and then reached 90% accuracy after just six sets of observations and the whole map converged after 15 sets of observations at 94% accuracy.

We consider these trials as early demonstrations of the soundness of the basic concept. In future, we will identify and adapt a robust 3D reconstruction technology, and run these experiments on *real world* objects within *real world* settings. We have skimmed over many of the details of auto-associative learning with self-organizing maps because we expect to make dramatic changes when we fine-tune the technique to real world problems. We are, however, extremely encouraged by the success of our relatively naïve implementation. In future, we plan to extend the use of SOMs to assist in constructing the graph-based models (*i.e.*, by extrapolating the obscured shape of the object from observed shape) and for allowing rich shape-based and affordance-based indexing, retrieval and similarity testing.

Performance Considerations

While placing an ordering on branches enables the tableau algorithm to emphasise optimal consistent branches and discontinue search on suboptimal branches, the suboptimal branches cannot be discarded from memory. This is because an optimal branch may later be found inconsistent, and therefore cause a previously suboptimal branch to become the most optimal *consistent* branch that remains.

In many cases, however, it may be known that this cannot happen. For example, it may be known that any inconsistency will apply to all ordered branches, or it may be known that ordered branches will never contain inconsistency. In this case it is possible to introduce Prolog-style 'cuts' into the tableau: special terms to indicate that non-optimal branches may be dropped. Of course, care must be taken to ensure that cuts are genuinely free of unintended side-effects—that they are 'green cuts', to use the terminology of Prolog.

Another concern is that robots will need to simultaneously learn while engaged in action. If action and learning occurs in the same tableau, there is a need to prevent branching in decision-making from causing the same learning problem to repeat in multiple branches of the tableau. This can be solved by careful factoring: continuous online learning is performed in a separate tableau, but the contents of that tableau are implicitly read in *logical conjunction* with the contents of the primary action and decision making tableau.

Such optimizations have straightforward implementation, however we will provide full details in future publication.

Conclusion

Parameter-driven simulations are not only an effective mechanism for rich commonsense reasoning, but they lend themselves to rapid and autonomous acquisition. We have shown how models of learning can be elegantly incorporated into the Comirit framework (and potentially other tableau-based systems). The extended framework thus simulateously combines the effectiveness and efficiency of simulation with the ability for autonomous knowledge acquisition, and with the full power and generality of logical formalisms.

To date, we have demonstrated the system on simple (but useful and plausible) learning problems. Early experimental results are extremely encouraging: the system learns rapidly and with very few observations.

As a long term goal, we plan to have Comirit autonomously acquire from observation, even the fundamental laws of a simulation and the mechanisms for model building. In particular, we hope that interaction in a complex environment (such as the 3D world) may be interpreted as a problem of determining a hierarchical non-linear flow of 'entities' within an environment. To whatever degree such automation is possible, our framework can accommodate any learning that can be expressed as an optimization problem, and yet allow for ongoing integration with symbolic and logical formalizations.

References

Gardin, G. and Meltzer, B. (1989) 'Analogical representations of naïve physics', *Artificial Intelligence*, vol. 38, no. 2, pp. 139–159.

Hähnle, R. (2001) 'Tableaux and Related Methods', *Handbook of Automated Reasoning*, vol. I, pp. 100–178, Elsevier Science.

Johnston, B. and Williams, M-A. (2007) 'A generic framework for approximate simulation in commonsense reasoning systems', *Proceedings of COMMONSENSE 2007*, pp. 71–76.

Johnston, B. and Williams, M-A. (2008) 'Comirit: Commonsense reasoning by integrating simulation and logic', *Proceedings of AGI-2008*, pp. 200–211.

Panton, K., Matuszek, C., Lenat, D., Schneider, D., Witbrock, M., Siegel, N. and Shepard, B. (2006) 'Common Sense Reasoning – From Cyc to Intelligent Assistant', In Yang Cai and Julio Abascal (eds.), *Ambient Intelligence in Everyday Life*, pp. 1-31, LNAI 3864, Springer, 2006.

An F-Measure for Context-based Information Retrieval

Michael Kandefer and Stuart Shapiro
Department of Computer Science and Engineering, Center for Cognitive Science,
Center for Multisource Information Fusion
University at Buffalo, Buffalo, NY 14260
{mwk3,shapiro}@cse.buffalo.edu

Abstract

Computationally expensive processes, such as deductive reasoners, can suffer performance issues when they operate over large-scale data sets. The optimal procedure would allow reasoners to only operate on that information that is relevant. Procedures that approach such an ideal are necessary to accomplish the goal of commonsense reasoning, which is to endow an agent with enough background knowledge to behave intelligently. Despite the presence of some procedures for accomplishing this task one question remains unanswered: How does one measure the performance of procedures that bring relevant information to bear in KR systems?

This paper answers this question by introducing two methods for measuring the performance of *context-based information retrieval* processes in the domain of KR systems. Both methods produce an *f-measure* as a result. These methods are evaluated with examples and discussion in order to determine which is more effective. Uses of these measures are also discussed.

Introduction

Computationally expensive processes, such as deductive reasoners, can suffer performance issues when they operate over large-scale data sets. The optimal procedure would allow such processes to operate with only the information that is relevant to the current task. Bringing relevant information to bear has numerous applications in context-aware agents/devices (Arritt & Turner 2003; Dey 2001; Bradley & Dunlop 2005; Dourish 2004; Kurz, Popescu, & Gallacher 2004).[1] In KR systems, reasoning is probably most hampered in large-scale knowledge bases due to complicated procedures, like building and maintaining search trees resulting from knowledge base queries. Such concerns with large-scale knowledge bases have been discussed previously (Subramanian, Greiner, & Pearl 1997) and various solutions have been offered (Haarslev & Moller 2001; Levy, Fikes, & Sagiv 1997; Lenat 1998; 1995). Due to these concerns, a method for including a minimal set of background knowledge for the current task is necessary if we are to accomplish the goals of commonsense reasoning, which is to endow an agent with all the commonsense knowledge necessary to exhibit intelligent behavior. Methods for solving this problem have been proposed or implemented in the past (Anderson 2007), and others are capable of being implemented in KR systems (Arritt & Turner 2003). However, one question remains unanswered: How does one measure the performance of procedures that bring relevant information to bear in KR?

This paper answers this question by discussing two methods for measuring the performance of *context-based information retrieval* (CBIR) processes in the domain of KR systems:

- Relevance Theoretic Measure, and
- Distance from the Optimal

The first is based on a method for determining the relevance of a subset of an agent's knowledge base given contextual information. Sperber and William (1995) initially proposed the *relevance-theoretic* approach for use in modeling communication, but believe it has uses in other cognitive processes. Harter (1992) agrees with this notion, but claims that the approach is also useful for determining relevance in information retrieval (IR) testing. Borlund and Ingwersern (1997) agree, but limit this type of testing to a particular type of relevance called "situated relevance". The *distance from the optimal* is our own method.

There are several uses for measurement methods like the above. The foremost is comparing the results of various CBIR procedures. Cohen (1995) has noted that we often do not know if a program has worked well, or poorly. Such evaluations often deal with speed and space considerations, but in CBIR procedures we are also interested in measuring the utility of the results. The measures above are one way of accomplishing this.

Other than comparing CBIR procedures, many CBIR procedures, such as spreading activation (Howes 2007; Crestani 1997; Loftus 1975) and context diagnosis (Arritt & Turner 2003), operate by utilizing various parameters that can be set to influence their performance. These parameters are given arbitrary values and then tested to find suitable levels. The above measurements schemes can aid in the process, and potentially make it automatic.

Though the two measurement methods can be used for

[1] Here "context" is not the knowledge representation and reasoning (KR) sense of the term, but defined as "the structured set of variable, external constraints to some (*real* or *artificial*) cognitive process that influences the behavior of that process in the agent(s) under consideration" (Kandefer & Shapiro 2008).

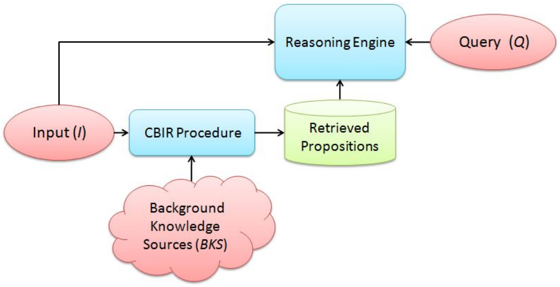

Figure 1: Context-Based Information Retrieval Process

the above tasks, our interest in this paper is in determining which makes a more effective tool for evaluating CBIR results. In order to accomplish this we will calculate the *f-measure* values of these methods when applied to example CBIR results. A *f-measure* is the standard measure for evaluating IR results.

Context-Based Information Retrieval

CBIR is an independent, preprocessing step that occurs before reasoning. A general CBIR procedure operates by examining an input, typically sensory. It uses that input to constrain the knowledge that is available to the reasoner. This process is depicted in Fig. 1.

The CBIR procedure receives input (*I*), which contains the *contextual constraints* and other information about the situation; and the background knowledge (*BKS*) containing any knowledge that will be evaluated by the CBIR procedure. With this the CBIR procedure produces a subset of the background knowledge, called *retrieved propositions*, for use by a reasoning engine, that can then be queried (*Q*). These queries could also be expected goals an embodied agent should be capable of achieving in context.

In "The Handle Problem" domain (Miller & Morgenstern 2006), which is the problem of inferring whether or not an object can be used as a handle through a description of its properties and relationships with other objects, an example of such input would be spatial information about some objects that might be door handles. Such information could contain a unique identifier for an unidentified object, the object's shape (e.g., conical, rectangular, etc.), and feature information (e.g., whether the object is inverted, or blunt). An example of background knowledge in such a domain would include various assertions about using objects with certain properties as handles. Though the CBIR procedure ultimately produces information for consumption by the reasoner, the tasks of the reasoner can also influence what information should be retrieved. This is apparent in goal seeking situations, such as question answering. As such, some CBIR procedures take into account the goals of the agent or the state of a problem they are solving and provide these as input.

The most important aspect of the process requires that the CBIR procedure output any *retrieved propositions*, which will be used by the reasoning engine. As such, the knowledge provided as relevant by the CBIR procedure will always be a subset of the *BKS*. This information is selected by the CBIR procedure through an algorithm that examines the *BKS* and *I*. This algorithm varies between CBIR procedures, but it should be noted that most do not examine the entirety of the *BKS*, but only an initial subset determined by *I*. The *retrieved propositions* determines a successful CBIR procedure, and what we will evaluate.

Measuring Results

As previously mentioned the output of the CBIR procedure is a subset of the *BKS*, called the *retrieved propositions*, and a means of establishing successful results is required. In information retrieval (IR) the accepted practice for evaluating such results is to calculate an *f-measure*. An *f-measure* score is between 0.0 and 1.0, with 0.0 indicating the poorest result and 1.0 a perfect retrieval. An *f-measure* identifies situations where IR results contain unnecessary information, called *precision*, and where the results do not contain enough information, called *recall*. In order to calculate an *f-measure* (Fig. 2) for CBIR results the *retrieved propositions* and another set of propositions, called the *relevant propositions*, are necessary. Below two methods for acquiring a set of *relevant propositions* and using them for evaluating the *retrieved propositions* from a CBIR process are discussed. In both of these methods the process of generating the set of *relevant propositions* can be accomplished any time prior to the calculation of the *f-measure* for the CBIR results.

Relevancy Theory

Relevancy Theory (Sperber & Wilson 1995) is a model developed by Wilson and Sperber in the field of pragmatics that is used for explaining the cognitive process listeners undertake as they approach an understanding of a speaker's utterance. A system implementing this model is said to be using a *relevance-theoretic* method. The *relevance-theoretic* approach is not limited to establishing the relevance of an utterance, but also of observable phenomena, memories, and current thoughts. The process of determining relevancy relies on a principle that states something is relevant to a cognitive agent, if the agent can utilize it to draw conclusions that matter to it. When such conclusions are reached this is said to be a *positive cognitive effect*.

In relevance theory these *positive cognitive effects* are utilized to *measure* the degree of relevancy of a particular input, where an input could be any of the cognitive artifacts discussed above. However, most of the discussion by Wilber and Sperber has focused on the communication aspects of relevancy, and determining when an utterance is relevant to the current working memory contents of an agent. The working memory of the agent is represented as a set of propositions, which are a subset of the contents of the agent's BKS. These assumptions are used by William and Sperber to define *positive cognitive effects*. We take that definition, but modify it slightly so it can be used for determining the set of relevant propositions in an agent's BKS, rather than an utterance the agent encounters. We

Recall (r)	**Precision (p)**	**F-measure (F)**
$r = \frac{\lvert\{relevant\ propositions\}\cap\{retrieved\ propositions\}\rvert}{\lvert\{relevant\ propositions\}\rvert}$	$p = \frac{\lvert\{relevant\ propositions\}\cap\{retrieved\ propositions\}\rvert}{\lvert\{retrieved\ propositions\}\rvert}$	$F(r,p) = \frac{2rp}{r+p}$

Figure 2: Formulas for computing the *f-measure* (van Rijsbergen 1979)

Entire Knowledge Base
$A1 : \forall(x,y)(Blunt(x) \wedge Conical(x) \wedge Drawer(y) \wedge ConnectedByTip(x,y) \rightarrow Handle(x)).$ $A2 : \forall(x)(Handle(x) \rightarrow CanBePulled(x)).$ $A3 : Blunt(h1).$ $A4 : Conical(h1).$ $A5 : \forall(x,y)(Rope(x) \wedge Light(y) \wedge Connected(x,y) \rightarrow CanBePulled(x)$ $A6 : \forall(x,y)(Blunt(x) \wedge Conical(y) \wedge ConnectedByBase(x,y) \rightarrow \neg Handle(x)$ $A7 : \forall(x)(Drawer(x) \rightarrow ContainsItems(x)).$

Figure 3: The entire knowledge base

define a *positive cognitive effect* as follows (italicized words will be discussed below):

Given I and *Q*, as sets of propositions, and *BKS*, then if there is a proposition *p* that is an element of BKS, but not an element of $\{I \cup Q\}$, then p is a *positive cognitive effect* if either:

1. $\neg p \in \{I \cup Q\}$,
2. *p* helps *strengthens* some *q* that is an element of $\{I \cup Q\}$, or
3. *p* contributes to a *contextual implication*, which is defined as the condition where:
   (a) $\{\{I \cup Q\} \cup BKS\}$ *non-trivially* derives using p some proposition *q*, and
   (b) $\{I \cup Q\}$ alone does not *non-trivially* derive *q*, and
   (c) BKS alone does not *non-trivially* derive *q*

In case (1) a comparison between the $\{I \cup Q\}$ and BKS is made that determines if any of the propositions contradict one another. Each proposition in BKS that does is considered a *positive cognitive effect*. Case (2) involves a notion of strengthening that can occur when two sets of propositions are compared. The strengthening of proposition q in $\{I \cup Q\}$ occurs when: (1) $\{\{I \cup Q\} \cup BKS\}$ *non-trivially* derives q, or (2) BKS *non-trivially* derives q, which was derived in $\{I \cup Q\}$ already. Any propositions that are members of BKS and that are involved in such derivations are counted as *positive cognitive effects*. Case (3) establishes as *positive cognitive effects* those propositions in BKS that are involved in a *non-trivial* derivation using propositions from both $\{I \cup Q\}$ and BKS, which can not be done by $\{I \cup Q\}$ or BKS independently.

Of the three cases two rely on a notion of a *non-trivial* derivation. A formalization of this concept is not trivial, not provided by William and Sperber, and beyond the scope of this paper. For the sake of simplicity we will consider any proposition involved in a *modus ponens* rule of inference to be *non-trivial* in our examples.

With the above method for establishing *positive cognitive effects* the *relevance-theoretic* approach can be used for measuring the relevancy of the set of *retrieved propositions* from a CBIR procedure. To accomplish this the above method is used to find all the *positive cognitive effects* in BKS. This resulting proposition set is taken as the *relevant propositions*. With the *retrieved propositions* and *relevant propositions* available the *recall*, *precision*, and *f-measure* can be calculated for each CBIR output using the formulas in Fig. 2.

To illustrate, assume we have the KB depicted in Fig. 3 as the BKS (created by us from propositions that might be useful for solving "The Door Handle Problem") and three retrieved proposition sets: *Usable Conical Drawer Handles*, *Conical Drawer Handles*, and *Misc. Handles and Drawers*. Assume also that these were output as relevant from three different CBIR procedures, and that they have the propositional content depicted in Fig. 4.

Suppose the following proposition set is a combination of the expected input and query, $\{I \cup Q\}$, to the agent in context: $\{Drawer(d1) \wedge ConnectedByTip(h1, d1) \wedge CanBePulled(h1)\}$. With this the *relevance theoretic* approach determines that $\{A1, A2, A3, A4, A7\}$ are the *relevant propositions* of the background knowledge sources as they are involved in part of *contextual implications* that result in the derivation of *Handle(h1)*, *CanBePulled(h1)* , and *ContainsItems(d1)*. With this the *f-measure* can be calculated for each CBIR *retrieved propositions* set. This is done using the cardinality of the *retrieved propositions* (Ret.), the cardinality of the *relevant propositions* (Rel.), and the cardinality of their intersection (Int.). (Fig. 5). For example, the *Usable Conical Drawer Handles* has retrieved four propositions that are all in the relevant proposition set. As such, the intersection between he two is also four and it receives a *precision* of 1.0 (4/4). However, there are five relevant propositions, as such the *recall* is 0.8 (4/5).

Given the results of the *f-measure* calculation, the retrieved proposition sets that is most relevant would be *Usable Conical Drawer Handles*. As such the relevancy method that retrieved that proposition set would be deemed better at the CBIR procedure than the other two.

Usable Conical Drawer Handle

$A1 : \forall(x,y)(Blunt(x) \wedge Conical(x) \wedge Drawer(y) \wedge ConnectedByTip(x,y) \rightarrow Handle(x)).$
$A2 : \forall(x)(Handle(x) \rightarrow CanBePulled(x)).$
$A3 : Blunt(h1).$
$A4 : Conical(h1).$

Conical Drawer Handles

$A1 : \forall(x,y)(Blunt(x) \wedge Conical(x) \wedge Drawer(y) \wedge ConnectedByTip(x,y) \rightarrow Handle(x)).$
$A2 : \forall(x)(Handle(x) \rightarrow CanBePulled(x)).$
$A3 : Blunt(h1).$
$A4 : Conical(h1).$
$A6 : \forall(x,y)(Blunt(x) \wedge Conical(y) \wedge ConnectedByBase(x,y) \rightarrow \neg Handle(x)$

Misc. Handles and Drawers

$A2 : \forall(x)(Handle(x) \rightarrow CanBePulled(x)).$
$A3 : Blunt(h1)$
$A4 : Conical(h1).$
$A5 : \forall(x,y)(Rope(x) \wedge Light(y) \wedge Connected(x,y) \rightarrow CanBePulled(x)$
$A7 : \forall(x)(Drawer(x) \rightarrow ContainsItems(x)).$

Figure 4: Three different outputs from different context-sensitive retrieval operations.

Retrieved Proposition Set	Rel.	Ret.	Int.	Recall	Precision	F-Measure
Usable Conical Drawer Handles	5	4	4	0.8	1.0	0.889
Conical Drawer Handles	5	5	4	0.8	0.8	0.8
Misc. Handles and Drawers	5	5	4	0.8	0.8	0.8

Figure 5: The results of calculating the *f-measure* using a relevance theoretic approach.

Distance from the Optimal

Distance from the optimal is a method of testing that examines the input to a system and creates the optimal results on which to compare a system's future performance. In the CBIR model, if given a reasoning query (i.e., a particular reasoning task given to the reasoner) *Q*, the input propositions *I*, the contents of the background knowledge sources *BKS*, and a reasoner, that is capable of keeping track of the origin sets,[2] or equivalent, then the optimal solution for the original query can be calculated. This is accomplished by the following algorithm:

1. Given some query proposition Q that the reasoner is asked to derive, the entire knowledge base BKS that the CBIR procedure would access, and an input I that the CBIR procedure would use to produce its output.
2. Load the BKS into the reasoner.
3. Add I to the BKS.
4. Query the reasoner on Q.
5. Examine the origin set for Q, OS_Q, defined as:[3]

$$OS_Q = \{A - I | A \subset \{BKS \cup I\} \wedge A \vdash Q \wedge \neg\exists(A')(A' \subsetneq A \wedge A' \vdash Q)\}.$$[4]

6. Select the sets in OS_Q that have the minimal cardinality. This new set of origin sets will be denoted with $min(OS_Q)$.[5]

After this process is complete we have those origin sets that derive Q, and that also contain the minimal number of propositions needed to do so. Since these propositions are necessary for reasoning to the desired conclusion and minimal, we shall consider any origin set in the set of minimal solutions an *optimal solution*. With the possible optimal solutions in hand, we can measure the results of a CBIR procedure against each optimal solution and compute a *f-measure* for the results.

The presence of multiple optimal solutions poses some problems for computing the *f-measure*. To handle this *recall*, *precision*, and *f-measure* must be calculated treating each optimal solution as the *relevant propositions* and then comparing it to the CBIR output, or the *retrieved propositions*. The highest *f-measure* is chosen as the result. The reason for choosing the highest is that the CBIR output might share few propositions with some of the optimal solutions, but still match one of them precisely. In such a scenario the CBIR

[2] An origin set for a proposition is the set of propositions used in the derivation of that proposition. Origin sets originate from *relevance logic* proof theory (Shapiro 1992).

[3] $A \vdash B$ indicates that a proposition B can be derived from the set of propositions A.

[4] *I* is removed since in a CBIR procedure it is automatically provided to the reasoner and it should not impact retrieval scores.

[5] This step is performed as there can be multiple reasoning "paths" to Q in a BKS that use different proposition sets.

Retrieved Proposition Set	Rel.	Ret.	Int.	Recall	Precision	F-Measure
Usable Conical Drawer Handles	4	4	4	1.0	1.0	1.0
Conical Drawer Handles	4	5	4	1.0	0.8	0.889
Misc. Handles and Drawers	4	5	3	0.75	0.6	0.667

Figure 6: The results of calculating the *f-measure* using the distance from the optimal approach.

output is at least capable of generating one of the perfect solutions. Formulas for *recall*, *precision*, and the *f-measure* are the same as those used in the *relevance-theoretic* approach (Fig. 2).

To illustrate how this measure can be used for evaluating the results of CBIR procedures consider an example using the the knowledge base depicted in Fig. 3 as the BKS parameter in the above algorithm. Let I be the proposition: $ConnectedByTip(h1, d1) \wedge Drawer(d1)$ and Q the query $CanBePulled(h1)$?.[6] After execution of the query we receive one origin set for Q: $\{A1, A2, A3, A4\}$, and since it is the only one, it is inserted into $min(OS_Q)$. With these values calculated we can now compare the optimal solution against the CBIR procedure outputs. We will again use the ones discussed in Fig. 4. The results are depicted in Fig. 6.

Since *Usable Conical Drawer Handles* is the actual optimal solution OS_Q it gets a perfect *f-measure* of 1.0. The *Conical Drawer Handles* receives the next highest as it had a perfect *recall*, but contain one extraneous proposition affecting its *precision*. The last retrieved proposition set, *Rope Handles*, was penalized heavily as it did not retrieve all of the relevant propositions (*recall*) and contained numerous propositions that weren't part of the relevant proposition set (*precision*).

Evaluation

Though both methods measure the same unit (i.e., propositions) and rely on rules of inference to ultimately create a score for the results, they differ in their generation of the *relevant propositions*, and thus, the *f-measure*. The greatest difference is that the *relevance-theoretic* uses the input to find all possible propositions that trigger *positive cognitive effects*, while the distance from the optimal only looks for the minimal set needed. This can result in needed discrimination when measuring CBIR results. In the example this is illustrated when the *relevance-theoretic* approach provided the same score for *Conical Drawer Handles* and *Misc. Handles and Drawers*, while the *distance from the optimal* provides a useful distinction.

The *relevance-theoretic* approach also values higher those CBIR outputs that contain multiple solutions to the same problem, since all propositions involved in those solutions would cause positive cognitive effects, despite the fact that only one solution is needed. This ultimately causes more reasoning and more computation time, which is what we would like a CBIR procedure to avoid. The *distance from the optimal* method values CBIR procedures that produce close to optimal solutions, and ones with multiple solutions would be considered as having extraneous propositions.

Finally, one important difference between the two methods is that the *relevance-theoretic* approach requires a formalization of a *non-trivial* deduction. This is not an easy task, as it involves determining which rules of inference and which combinations of them are trivial.

Measurement Requirements

For the two methods presented for measuring CBIR in knowledge representation and reasoning (KR) a KR system is needed to perform the actual measurements. This system need not be the same as the one in the process diagram. It requires the ability to:

- **Store and reason over a large number of propositions.** The CBIR methods are designed to retrieve relevant information from larger knowledge bases and present them to a reasoner to limit processing. While this design is done to eliminate the need for a KR system to have all of the background information available to it, measuring the success of CBIR processes does need a KR system to reason over the entire knowledge base every time a *relevant proposition* set needs to be generated. While the term "large" is vague a knowledge base with approximately 100,000 propositions causes problems for some reasoning tasks. Speed of reasoning is not required for the measurements.
- **Perform forward and backward chaining.** Both measurements rely on forward chaining, and the *distance from the optimal* relies on backward chaining to generate the set of *relevant propositions*, which are used in measuring the results of the CBIR procedures.
- **Detect non-trivial deductions.** The *relevance-theoretic* approach requires that the KR system recognize *non-trivial* derivations in order to prevent the mislabeling of trivial propositions in that derivation as relevant. These *non-trivial* derivations are needed to populate the set of *relevant propositions*.
- **Compute and store the origin sets of derived propositions.** The *distance from the optimal* measurement requires that the KR system keep track of the minimal number of propositions required to derive another proposition (i.e., the origin set) in order to create the set of *relevant propositions*.

Conclusions and Future Work

The *relevance-theoretic* approach for determining the relevant propositions in a knowledge base, an approach that has been proposed as a useful method for determining relevancy

[6] This differs slightly from the previous example since the *relevance theoretic* approach does not take into account how the retrieved propositions will be used (e.g., expected queries, agent goals).

in information retrieval, was found to be less successful than the *distance from the optimal* method for measuring the results of CBIR procedures. This was mostly because the *relevant theoretic* approach finds all solutions to a problem and marks all propositions involved in those solutions as relevant, while the *distance from the optimal* finds the minimal number. Some theoretical issues also hinder the *relevance-theoretic* approach. Its reliance on *trivial implications* is one such issue, as they are difficult to properly formalize. This formalization step is necessary prior to development of a tool that can use the *relevance-theoretic* approach for measuring the results of CBIR procedures. The long-term goals of commonsense reasoning will require methods for retrieving a subset of an agent's background knowledge based on context. CBIR procedures address this issue, and choosing method for measuring their performance will be required.

Apart from these findings, a theoretical discussion and an example was used to compare the two measures. Examples like this serve as a useful precursor to the development of test cases for evaluating the measures against each other. In the future we will explore such test cases. In doing so, a formalization of *non-trivial* deductions will also be produced.

References

Anderson, J. R. 2007. Human associative memory. In *How Can the Human Mind Occur in the Physical Universe?* NY, New York: Oxford University Press. 91–134.

Arritt, R., and Turner, R. 2003. Situation assessment for autonomous underwater vehicles using a priori contextual knowledge. In *Proceedings of the Thirteenth International Symposium on Unmanned Untethered Submersible Technology (UUST)*.

Borlund, P., and Ingwersern, P. 1997. The development of a method for the evaluation of interactive information retrieval systems. *The Journal of Documentation* 53(3).

Bradley, N. A., and Dunlop, M. D. 2005. Toward a multidisciplinary model of context to support context-aware computing. *Human-Computer Interaction* 20:403–446.

Cohen, P. R. 1995. Empirical research. In *Empirical Methods for Artificial Intelligence*. The MIT Press,Cambridge, MA.

Crestani, F. 1997. Application of spreading activation techniques in information retrieval. *Artif. Intell. Rev.* 11(6):453–482.

Dey, A. K. 2001. Understanding and using context. *Personal and Ubiquitous Computing* 5(1):4–7.

Dourish, P. 2004. What we talk about when we talk about context. *Personal Ubiquitous Computing* 8(1):19–30.

Haarslev, V., and Moller, R. 2001. High performance reasoning with very large knowledge bases: A practical case study. In *International Joint Conference on Artificial Intelligence*.

Harter, S. P. 1992. Psychological relevance and information science. *Journal of the American Society for Information Science* 43(9).

Howes, M. B. 2007. *Human Memory*. Sage Publications, Inc. chapter Long-term Memory – Ongoing Research, 131–160.

Kandefer, M., and Shapiro, S. C. 2008. A categorization of contextual constraints. In Samsonovich, A., ed., *Biologically Inspired Cognitive Architectures, Technical Report FS–08–04*, 88–93. AAAI Press, Menlo Park, CA.

Kurz, B.; Popescu, I.; and Gallacher, S. 2004. FACADE - a framework for context-aware content adaptation and delivery. In *Proceedings of Communication Networks and Services Research*, 46–55.

Lenat, D. B. 1995. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM* 38(11).

Lenat, D. 1998. The Dimensions of Context Space. Technical Report, Cycorp.

Levy, A. Y.; Fikes, R. E.; and Sagiv, Y. 1997. Speeding up inferences using relevance reasoning: a formalism and algoirthms. *Artificial Intelligence* 97(1–2):pg. 83–136.

Loftus, A. C. E. 1975. A spreading activation theory of semantic processing. *Psychological Review* 82(6):407–428.

Miller, R., and Morgenstern, L. 2006. Common sense problem page. `http://www-formal.stanford.edu/leora/commonsense/`.

Shapiro, S. C. 1992. Relevance Logic in Computer Science. In Anderson, A. R.; Nuel D. Belnap, J.; and Dunn, J. M., eds., *Entailment*, volume II. Princeton, NJ: Princeton University Press. pg. 553–563.

Sperber, D., and Wilson, D. 1995. *Relevance: Communication and Cognition*. Blackwell Publishing, Malden, MA.

Subramanian, D.; Greiner, R.; and Pearl, J. 1997. The relevance of relevance. *Artificial Intelligence* 97(1–2):pg. 1–5.

van Rijsbergen, C. 1979. *Information Retrieval, Second Edition*. Butterworths.

A Logical Account of Prioritized Goals and their Dynamics

Shakil M. Khan and Yves Lespérance
Department of Computer Science and Engineering
York University, Toronto, ON, Canada
{skhan, lesperan}@cse.yorku.ca

Abstract

Most previous logical accounts of goals do not deal with prioritized goals and goal dynamics properly. Many are restricted to achievement goals. In this paper, we develop a logical account of goal change that addresses these deficiencies. In our account, we do not drop lower priority goals permanently when they become inconsistent with other goals and the agent's knowledge; rather, we make such goals inactive. We ensure that the agent's chosen goals/intentions are consistent with each other and the agent's knowledge. When the world changes, the agent recomputes her chosen goals and some inactive goals may become active again. This ensures that our agent maximizes her utility. We prove that the proposed account has desirable properties.

Introduction

There has been much work on modeling agent's mental states, beliefs, goals, and intentions, and how they interact and lead to rational decisions about action. As well, there has been a lot of work on modeling belief change. But the dynamics of motivational attitudes has received much less attention. Most formal models of goal and goal change (Cohen and Levesque 1990; Rao and Georgeff 1991; Konolige and Pollack 1993; Shapiro *et al.* 1995) assume that all goals are equally important and many only deal with achievement goals. Moreover, most of these frameworks do not guarantee that an agent's goals will properly evolve when an action/event occurs, e.g. when the agent's beliefs/knowledge changes or a goal is adopted or dropped (one exception to this is the model of prioritized goals in (Shapiro and Brewka 2007)). Dealing with these issues is important for developing effective models of rational agency. It is also important for work on BDI agent programming languages, where handling declarative goals is an active research topic.

In this paper, we present a formal model of prioritized goals and their dynamics that addresses some of these issues. In our framework, an agent can have multiple goals at different priority levels, possibly inconsistent with each other. We define intentions as the maximal set of highest priority goals that is consistent given the agent's knowledge. Our model of goals supports the specification of general temporally extended goals, not just achievement goals.

We start with a (possibly inconsistent) initial set of *prioritized goals* or desires that are totally ordered according to priority, and specify how these goals evolve when actions/events occur and the agent's knowledge changes. We define the agent's *chosen goals* or intentions in terms of this goal hierarchy. Our agents maximize their utility; they will abandon a chosen goal ϕ if an opportunity to commit to a higher priority but inconsistent with ϕ goal arises. To this end, we keep all prioritized goals in the goal base unless they are explicitly dropped. At every step, we compute an optimal set of chosen goals given the hierarchy of prioritized goals, preferring higher priority goals such that chosen goals are consistent with each other and with the agent's knowledge. Thus at any given time, some goals in the hierarchy are active, i.e. chosen, while others are inactive. Some of these inactive goals may later become active, e.g. if a higher priority active goal that is currently blocking an inactive goal becomes impossible.

Our formalization of prioritized goals ensures that the agent always tries to maximize her utility, and as such displays an idealized form of rationality. In the fifth section, we discuss how this relates to Bratman's (1987) theory of practical reasoning. We use an action theory based on the situation calculus along with our formalization of paths in the situation calculus as our base formalism.

In the next section, we outline our base framework. In the third section, we formalize *paths* in the situation calculus to support modeling goals. In the fourth section, we present our model of prioritized goals. In the fifth and sixth section, we present our formalization of goal dynamics and discuss some of its properties. Then in the last section, we summarize our results, discuss previous work in this area, and point to possible future work.

Action and Knowledge

Our base framework for modeling goal change is the situation calculus as formalized in (Reiter 2001). In this framework, a possible state of the domain is represented by a situation. There is a set of initial situations corresponding to the ways the agent believes the domain might be initially, i.e. situations in which no actions have yet occurred. $\mathrm{Init}(s)$ means that s is an initial situation. The actual initial state is represented by a special constant S_0. There is a distinguished binary function symbol *do* where $do(a, s)$ denotes the successor situation to s resulting from performing the action a. Relations (and functions) whose truth values vary from situation to situation, are called relational (functional, resp.) fluents, and are denoted by predicate (function, resp.) symbols taking a situation term as their last argument. There is a special predicate $\mathrm{Poss}(a, s)$ used to state that action a is executable in situation s.

Our framework uses a theory D_{basic} that includes the following set of axioms:[1] (1) action precondition axioms, one per action a characterizing $\mathrm{Poss}(a, s)$, (2) successor state axioms (SSA), one per fluent, that succinctly encode both effect and frame axioms and specify exactly when the fluent changes (Reiter 2001), (3) initial state axioms describing what is true initially including the mental states of the agents, (4) unique name axioms for actions, and (5) domain-independent foundational axioms describing the structure of situations (Levesque *et al.* 1998).

Following (Scherl and Levesque 2003), we model knowledge using a possible worlds account adapted to the situation calculus. $K(s', s)$ is used to denote that in situation s, the agent thinks that she could be in situation s'. Using K, the knowledge of an agent is defined as:[2] $\mathrm{Know}(\Phi, s) \overset{\text{def}}{=} \forall s'.\ K(s', s) \supset \Phi(s')$, i.e. the agent knows Φ in s if Φ holds in all of her K-accessible situations in s. K is constrained to be reflexive, transitive, and Euclidean in the initial situation to capture the fact that agents' knowledge is true, and that agents have positive and negative introspection. As shown in (Scherl and Levesque 2003), these constraints then continue to hold after any sequence of actions since they are preserved by the successor state axiom for K. We also assume that all actions are public, i.e. whenever an action (including exogenous events) occurs, the agent learns that it has happened. Note that, we work with knowledge rather than belief. Although much of our formalization should extend to the latter, we leave this for future work.

Paths in the Situation Calculus

To support modeling temporally extended goals, we introduce a new sort of *paths*, with (possibly sub/super-scripted) variables p ranging over paths. A path is essentially an infinite sequence of situations, where each situation along the path can be reached by performing some *executable* action in the preceding situation. We introduce a predicate $\mathrm{OnPath}(p, s)$, meaning that the situation s is on path p. Also, $\mathrm{Starts}(p, s)$ means that s is the starting situation of path p. A path p starts with s iff s is the earliest situation on p:[3]

Axiom 1

$$\mathrm{Starts}(p, s) \equiv \mathrm{OnPath}(p, s) \wedge \forall s'.\ \mathrm{OnPath}(p, s') \supset s \le s'.$$

In the standard situation calculus, paths are implicitly there, and a path can be viewed as a pair (s, F) consisting of a situation s representing the starting situation of the path, and a function F from situations to actions (called *Action Selection Functions* (ASF) or strategies in (Shapiro *et al.* 1995)), such that from the starting situation s, F defines an infinite sequence of situations by specifying an action for every situation starting from s. Thus, one way of axiomatizing paths is by making them correspond to such pairs (s, F):

Axiom 2 $\forall p.\ \mathrm{Starts}(p, s) \supset (\exists F.\ \mathrm{Executable}(F, s)$
$\wedge \forall s'.\ \mathrm{OnPath}(p, s') \equiv \mathrm{OnPathASF}(F, s, s'))$,
$\forall F, s.\ \mathrm{Executable}(F, s) \supset \exists p.\ \mathrm{Starts}(p, s)$
$\wedge \forall s'.\ \mathrm{OnPathASF}(F, s, s') \equiv \mathrm{OnPath}(p, s')$.

This says that for every path there is an executable ASF that produces exactly the sequence of situations on the path from its starting situation. Also, for every executable ASF and situation, there is a path that corresponds to the sequence of situations produced by the ASF starting from that situation.

$$\mathrm{OnPathASF}(F, s, s') \overset{\text{def}}{=}$$
$$s \le s' \wedge \forall a, s^*.\ s < do(a, s^*) \le s' \supset F(s^*) = a,$$

$$\mathrm{Executable}(F, s) \overset{\text{def}}{=} \forall s'.\ \mathrm{OnPathASF}(F, s, s') \supset \mathrm{Poss}(F(s'), s').$$

Here, $\mathrm{OnPathASF}(F, s, s')$ means that the situation sequence defined by (s, F) includes the situation s'. Also, the situation sequence encoded by a strategy F and a starting situation s is executable iff for all situations s' on this sequence, the action selected by F in s' is executable in s'.

We will use both state and path formulae. A state formula $\Phi(s)$ is a formula that has a free situation variable s in it, whereas a path formula $\phi(p)$ is one that has a free path variable p. State formulae are used in the context of knowledge while path formulae are used in that of goals. Note that, by incorporating infinite paths in our framework, we can evaluate goals over these and handle arbitrary temporally extended goals; thus, unlike some other situation calculus based accounts where goal formulae are evaluated w.r.t. finite paths (e.g. (Shapiro and Brewka 2007)), we can handle for example unbounded maintenance goals.

We next define some useful constructs. A state formula Φ *eventually holds* over the path p if Φ holds in some situation that is on p, i.e. $\Diamond\Phi(p) \overset{\text{def}}{=} \exists s'.\ \mathrm{OnPath}(p, s') \wedge \Phi(s')$. Other Temporal Logic operators can be defined similarly, e.g. always Φ: $\Box\Phi(p)$.

An agent *knows* in s that ϕ has become *inevitable* if ϕ holds over all paths that starts with a K-accessible situation in s, i.e. $\mathrm{KInevitable}(\phi, s) \overset{\text{def}}{=} \forall p.\ \mathrm{Starts}(p, s') \wedge K(s', s) \supset \phi(p)$. An agent knows in s that ϕ is impossible if she knows that $\neg\phi$ is inevitable in s, i.e. $\mathrm{KImpossible}(\phi, s) \overset{\text{def}}{=} \mathrm{KInevitable}(\neg\phi, s)$.

Thirdly, we define what it means for a path p' to be a suffix of another path p w.r.t. a situation s:

$$\mathrm{Suffix}(p', p, s) \overset{\text{def}}{=} \mathrm{OnPath}(p, s) \wedge \mathrm{Starts}(p', s)$$
$$\wedge \forall s'.\ s' \ge s \supset \mathrm{OnPath}(p, s') \equiv \mathrm{OnPath}(p', s').$$

Fourthly, $\mathrm{SameHist}(s_1, s_2)$ means that the situations s_1 and s_2 share the same history of actions, but perhaps starting from different initial situations:

Axiom 3 $\mathrm{SameHist}(s_1, s_2) \equiv (\mathrm{Init}(s_1) \wedge \mathrm{Init}(s_2)) \vee$
$(\exists a, s_1', s_2'.\ s_1 = do(a, s_1') \wedge s_2 = do(a, s_2') \wedge \mathrm{SameHist}(s_1', s_2'))$.

Finally, we say that ϕ has become *inevitable* in s if ϕ holds over all paths that starts with a situation that has the same history as s: $\mathrm{Inevitable}(\phi, s) \overset{\text{def}}{=} \forall p, s'.\ \mathrm{Starts}(p, s') \wedge \mathrm{SameHist}(s', s) \supset \phi(p)$.

[1] We will be quantifying over formulae, and thus assume D_{basic} includes axioms for encoding of formulae as first order terms, as in (Shapiro *et al.* 2007).

[2] Φ is a state formula that can contain a situation variable, *now*, in the place of situation terms. We often suppress *now* when the intent is clear from the context.

[3] In the following, $s < s'$ means that s' can be reached from s by performing a sequence of executable actions. $s \le s'$ is an abbreviation for $s < s' \vee s = s'$.

Prioritized Goals

Most work on formalizing goals only deals with static goal semantics and not their dynamics. In this section, we formalize goals or desires with different priorities which we call *prioritized goals* (p-goals, henceforth). These p-goals are not required to be mutually consistent and need not be actively pursued by the agent. In terms of these, we define the consistent set of *chosen goals* or intentions (c-goals, henceforth) that the agent is committed to. In the next section, we formalize goal dynamics by providing a SSA for p-goals. The agent's c-goals are automatically updated when her p-goals change.

Not all of the agent's goals are equally important to her. Thus, it is useful to support a priority ordering over goals. This information can be used to decide which of the agent's c-goals should no longer be actively pursued in case they become mutually inconsistent. We specify each p-goal by its own accessibility relation/fluent G. A path p is G-accessible at priority level n in situation s (denoted by $G(p, n, s)$) if all the goals of the agent at level n are satisfied over this path and if it starts with a situation that has the same action history as s. The latter requirement ensures that the agent's p-goal-accessible paths reflect the actions that have been performed so far. A smaller n represents higher priority, and the highest priority level is 0. Thus here we assume that the set of p-goals are totally ordered according to priority. We say that an agent has the p-goal that ϕ at level n in situation s iff ϕ holds over all paths that are G-accessible at n in s:

$$\mathrm{PGoal}(\phi, n, s) \overset{\mathrm{def}}{=} \forall p.\ G(p, n, s) \supset \phi(p).$$

To be able to refer to all the p-goals of the agent at some given priority level, we also define *only p-goals*.

$$\mathrm{OPGoal}(\phi, n, s) \overset{\mathrm{def}}{=} \mathrm{PGoal}(\phi, n, s) \wedge \forall p.\ \phi(p) \supset G(p, n, s).$$

An agent has the only p-goal that ϕ at level n in situation s iff ϕ is a p-goal at n in s, and any path over which ϕ holds is G-accessible at n in s.

A domain theory for our framework D includes the axioms of a theory D_{basic} as in the previous section, the axiomatization of paths i.e. axioms 1-3, domain dependent initial goal axioms (see below), the domain independent axioms 4-7 and the definitions that appear in this section and the next. We allow the agent to have infinitely many goals. We expect the modeler to include some specification of what paths are G accessible at the various levels initially. We call these axioms *initial goal axioms*. In many cases, the user will want to specify a finite set of initial p-goals. This can be done by providing a set of axioms as in the example below. But in general, an agent can have a countably infinite set of p-goals, e.g. an agent that has the p-goal at level n to know what the n-th prime number is for all n. The agent's set of p-goals can even be specified incompletely, e.g. the theory might not specify what the p-goals at some level are initially.

We use the following as a running example. We have an agent who initially has the following three p-goals: $\phi_0 = \Box\mathrm{BeRich}$, $\phi_1 = \Diamond\mathrm{GetPhD}$, and $\phi_2 = \Box\mathrm{BeHappy}$ at level 0, 1, and 2, respectively. This domain can be specified using the following two initial goal axioms:

$$\begin{aligned}
&\text{(a) } \mathrm{Init}(s) \supset ((G(p, 0, s) \equiv \mathrm{Starts}(p, s') \wedge \mathrm{Init}(s') \wedge \phi_0(p)) \\
&\quad \wedge ((G(p, 1, s) \equiv \mathrm{Starts}(p, s') \wedge \mathrm{Init}(s') \wedge \phi_1(p)) \\
&\quad \wedge (G(p, 2, s) \equiv \mathrm{Starts}(p, s') \wedge \mathrm{Init}(s') \wedge \phi_2(p))),
\end{aligned}$$

$$\begin{aligned}
&\text{(b) } \forall n, p, s.\ \mathrm{Init}(s) \wedge n \geq 3 \supset \\
&\qquad (G(p, n, s) \equiv \mathrm{Starts}(p, s') \wedge \mathrm{Init}(s')).
\end{aligned}$$

(a) specifies the p-goals ϕ_0, ϕ_1, ϕ_2 (from highest to lowest priority) of the agent in the initial situations, and makes $G(p, n, s)$ true for every path p that starts with an initial situation and over which ϕ_n holds, for $n = 0, 1, 2$; each of them defines a set of initial goal paths for a given priority level, and must be consistent. (b) makes $G(p, n, s)$ true for every path p that starts with an initial situation for $n \geq 3$. Thus at levels $n \geq 3$, the agent has the trivial p-goal that she be in an initial situation. Assume that while initially the agent knows that all of her p-goals are individually achievable, she knows that her p-goal $\Diamond$GetPhD is inconsistent with her highest priority p-goal $\Box$BeRich as well as with her p-goal $\Box$BeHappy, while the latter are consistent with each other. Thus in our example, we have $\mathrm{OPGoal}(\phi_i(p) \wedge \mathrm{Starts}(p, s) \wedge \mathrm{Init}(s), i, S_0)$, for $i = 0, 1, 2$. Also, for any $n \geq 3$, we have $\mathrm{OPGoal}(\mathrm{Starts}(p, s) \wedge \mathrm{Init}(s), n, S_0)$.

While p-goals or desires are allowed to be known to be impossible to achieve, an agent's c-goals or intentions must be realistic. Not all of the G-accessible paths are realistic in the sense that they start with a K-accessible situation. To filter these out, we define *realistic* p-goal accessible paths:

$$G_R(p, n, s) \overset{\mathrm{def}}{=} G(p, n, s) \wedge \mathrm{Starts}(p, s') \wedge K(s', s),$$

Thus G_R prunes out the paths from G that are known to be impossible, and since we define c-goals in terms of realistic p-goals, this ensures that c-goals are realistic. We say that an agent has the *realistic p-goal* that ϕ at level n in situation s iff ϕ holds over all paths that are G_R-accessible at n in s:

$$\mathrm{RPGoal}(\phi, n, s) \overset{\mathrm{def}}{=} \forall p.\ G_R(p, n, s) \supset \phi(p).$$

Using realistic p-goals, we next define c-goals. The idea of how we calculate c-goal-accessible paths is as follows: the set of G_R-accessibility relations represents a set of prioritized temporal propositions that are candidates for the agent's c-goals. Given G_R, in each situation we want to compute the agent's c-goals such that it is the *maximal consistent* set of higher priority realistic p-goals. We do this iteratively starting with the set of all realistic paths (i.e. paths that starts with a K-accessible situation). At each iteration we compute the intersection of this set with the next highest priority set of G_R-accessible paths. If the intersection is not empty, we thus obtain a new chosen set of paths at level i. We call a p-goal chosen by this process an *active* p-goal. If on the other hand the intersection is empty, then it must be the case that the p-goal represented by this level is either in conflict with another active higher priority p-goal/a combination of two or more active higher priority p-goals, or is known to be impossible. In that case, that p-goal is ignored (i.e. marked as inactive), and the chosen set of paths at level i is the same as at level $i - 1$. Axiom 4 computes this intersection:[4]

Axiom 4 $G_\cap(p, n, s) \equiv$

```
    if ($n = 0$) then
        if $\exists p'.\ G_R(p', n, s)$ then $G_R(p, n, s)$
        else $\mathrm{Starts}(p, s') \wedge K(s', s)$
```

[4] **if** ϕ **then** δ_1 **else** δ_2 is an abbreviation for $(\phi \supset \delta_1) \wedge (\neg\phi \supset \delta_2)$.

$$\begin{array}{l} \textbf{else} \\ \quad \textbf{if } \exists p'.(G_R(p', n-1, s) \wedge G_\cap(p', n-1, s)) \\ \qquad \textbf{then } (G_R(p, n-1, s) \wedge G_\cap(p, n-1, s)) \\ \quad \textbf{else } G_\cap(p, n-1, s). \end{array}$$

Using this, we define what it means for an agent to have a c-goal at some level n:

$$\text{CGoal}(\phi, n, s) \overset{\text{def}}{=} \forall p.\ G_\cap(p, n, s) \supset \phi(p),$$

i.e. an agent has the c-goal at level n that ϕ if ϕ holds over all paths that are in the prioritized intersection of the set of G_R-accessible paths up to level n.

We define c-goals in terms of c-goals at level n:

$$\text{CGoal}(\phi, s) \overset{\text{def}}{=} \forall n.\ \text{CGoal}(\phi, n, s),$$

i.e., the agent has the c-goal that ϕ if for any level n, ϕ is a c-goal at n.

In our example, the agent's realistic p-goals are $\Box$BeRich, $\Diamond$GetPhD, and $\Box$BeHappy in order of priority. The $G_\cap$-accessible paths at level 0 in S_0 are the ones that start with a K-accessible situation and where $\Box$BeRich holds. The $G_\cap$-accessible paths at level 1 in S_0 are the same as at level 0, since there are no K-accessible paths over which both $\Diamond$GetPhD and $\Box$BeRich hold. Finally, the $G_\cap$-accessible paths at level 2 in S_0 are those that start with a K-accessible situation and over which $\Box$BeRich $\wedge$ $\Box$BeHappy holds. Also, it can be shown that initially our example agent has the c-goals that $\Box$BeRich and $\Box$BeHappy, but not $\Diamond$GetPhD.

Note that by our definition of c-goals, the agent can have a c-goal that ϕ in situation s for various reasons: 1) ϕ is known to be inevitable in s; 2) ϕ is an active p-goal at some level n in s; 3) ϕ is a consequence of two or more active p-goals at different levels in s. To be able to refer to c-goals for which the agent has a primitive motivation, i.e. c-goals that result from a single active p-goal at some priority level n, in contrast to those that hold as a consequence of two or more active p-goals at different priority levels, we define *primary* c-goals:

$$\begin{array}{l} \text{PrimCGoal}(\phi, s) \overset{\text{def}}{=} \\ \qquad \exists n.\ \text{PGoal}(\phi, n, s) \wedge \exists p.\ G(p, n, s) \wedge G_\cap(p, n, s). \end{array}$$

That is, an agent has the primary c-goal that ϕ in situation s, if ϕ is a p-goal at some level n in s, and if there is a G-accessible path p at n in s that is also in the prioritized intersection of G_R-accessible paths upto n in s. The last two conjucts are required to ensure that n is an active level. Thus if an agent has a primary c-goal that ϕ, then she also has the c-goal that ϕ, but not necessarily vice-versa. It can be shown that initially our example agent has the primary c-goals that $\Box$BeRich and $\Box$BeHappy, but not their conjunction. To some extent, this shows that primary c-goals are not closed under logical consequence.

Goal Dynamics

An agent's goals change when her knowledge changes as a result of the occurrence of an action (including exogenous events), or when she adopts or drops a goal. We formalize this by specifying how p-goals change. C-goals are then computed using (realistic) p-goals in every new situation as above.

We introduce two actions for adopting and dropping a p-goal, $adopt(\phi, n)$ and $drop(\phi)$. The action precondition axioms for these are as follows:

Axiom 5 $\text{Poss}(adopt(\phi, n), s) \equiv \neg\exists n'.\ \text{PGoal}(\phi, n', s),$
$\text{Poss}(drop(\phi), s) \equiv \exists n.\ \text{PGoal}(\phi, n, s).$

That is, an agent can adopt (drop) the p-goal that ϕ at level n, if she does not (does) already have ϕ as her p-goal at some level.

In the following, we specify the dynamics of p-goals by giving the SSA for G and discuss each case, one at a time:

Axiom 6 (SSA for G) $G(p, n, do(a, s)) \equiv$

$$\begin{array}{l} \forall \phi, m.\ (a \neq adopt(\phi, m) \wedge a \neq drop(\phi) \wedge \text{Progressed}(p, n, a, s)) \\ \vee\ \exists \phi, m.\ (a = adopt(\phi, m) \wedge \text{Adopted}(p, n, m, a, s, \phi)) \\ \vee\ \exists \phi.\ (a = drop(\phi) \wedge \text{Dropped}(p, n, a, s, \phi)). \end{array}$$

The overall idea of the SSA for G is as follows. First of all, to handle the occurrence of a non-adopt/drop (i.e. regular) action a, we progress all G-accessible paths to reflect the fact that this action has just happened; this is done using the $\text{Progressed}(p, n, a, s)$ construct, which replaces each G-accessible path p' with starting situation s', by its suffix p provided that it starts with $do(a, s')$:

$$\begin{array}{l} \text{Progressed}(p, n, a, s) \overset{\text{def}}{=} \\ \qquad \exists p'.\ G(p', n, s) \wedge \text{Starts}(p', s') \wedge \text{Suffix}(p, p', do(a, s')). \end{array}$$

Any path over which the next action performed is not a is eliminated from the respective G accessibility level.

Secondly, to handle adoption of a p-goal ϕ at level m, we add a new proposition containing the p-goal to the agent's goal hierarchy at m. The G-accessible paths at all levels above m are progressed as above. The G-accessible paths at level m are the ones that share the same history with $do(a, s)$ and over which ϕ holds. The G-accessible paths at all levels below m are the ones that can be obtained by progressing the level immediately above it. Thus the agent acquires the p-goal that ϕ at level m, and all the p-goals with priority m or less in s are pushed down one level in the hierarchy.

$$\begin{array}{l} \text{Adopted}(p, n, m, a, s, \phi) \overset{\text{def}}{=} \\ \quad \textbf{if } (n < m) \textbf{ then } \text{Progressed}(p, n, a, s) \\ \quad \textbf{else if } (n = m) \textbf{ then } \exists s'.\ \text{Starts}(p, s') \\ \qquad\qquad\qquad\qquad \wedge\ \text{SameHist}(s', do(a, s)) \wedge \phi(p) \\ \quad \textbf{else } \text{Progressed}(p, n-1, a, s). \end{array}$$

Finally, to handle the dropping of a p-goal ϕ, we replace the propositions that imply the dropped goal in the agent's goal hierarchy by the trivial proposition that the history of actions in the current situation has occurred. Thus, in addition to progressing all G-accessible paths as above, we add back all paths that share the same history with $do(a, s)$ to the existing G-accessibility levels where the agent has the p-goal that ϕ.

$$\begin{array}{l} \text{Dropped}(p, n, a, s, \phi) \overset{\text{def}}{=} \textbf{if } \text{PGoal}(\phi, n, s) \\ \qquad \textbf{then } \exists s'.\ \text{Starts}(p, s') \wedge \text{SameHist}(s', do(a, s)) \\ \qquad \textbf{else } \text{Progressed}(p, n, a, s). \end{array}$$

Returning to our example, recall that our agent has the c-goals/active p-goals in S_0 that $\Box$BeRich and $\Box$BeHappy, but not $\Diamond$GetPhD, since the latter is inconsistent with her higher priority p-goal $\Box$BeRich. Assume that, after the action *goBankrupt* happens in S_0, the p-goal $\Box$BeRich becomes impossible. Then in $S_1 = do(goBankrupt, S_0)$, the agent has the c-goal that $\Diamond$GetPhD, but not $\Box$BeRich nor $\Box$BeHappy; $\Box$BeRich is excluded from the set of c-goals since it has become impossible to achieve (i.e. unrealistic). Also, since her higher priority p-goal $\Diamond$GetPhD is inconsistent with her p-goal $\Box$BeHappy, the agent will make $\Box$BeHappy inactive.

Note that, while it might be reasonable to drop a p-goal (e.g. $\Diamond$GetPhD) that is in conflict with another higher priority active p-goal (e.g. $\Box$BeRich), in our framework we keep such p-goals around. The reason for this is that although $\Box$BeRich is currently inconsistent with $\Diamond$GetPhD, the agent might later learn that $\Box$BeRich has become impossible to bring about (e.g. after *goBankrupt* occurs), and then might want to pursue $\Diamond$GetPhD. Thus, it is useful to keep these inactive p-goals since this allows the agent to maximize her utility (that of her chosen goals) by taking advantage of such opportunities. As mentioned earlier, c-goals are our analogue to intentions. Recall that Bratman's (1987) model of intentions limits the agent's practical reasoning – agents do not always optimize their utility and don't always reconsider all available options in order to allocate their reasoning effort wisely. In contrast to this, our c-goals are defined in terms of the p-goals, and at every step, we ensure that the agent's c-goals maximize her utility so that these are the set of highest priority goals that are consistent given the agent's knowledge. Thus, our notion of c-goals is not as persistent as Bratman's notion of intention. For instance as mentioned above, after the action *goBankrupt* happens in S_0, the agent will lose the c-goal that $\Box$BeHappy, although she did not drop it and it did not become impossible or achieved. In this sense, our model is that of an idealized agent. There is a tradeoff between optimizing the agent's chosen set of prioritized goals and being committed to chosen goals. In our framework, chosen goals behave like intentions with an automatic filter-override mechanism (Bratman 1987) that forces the agent to drop her chosen goals when opportunities to commit to other higher priority goals arise. In the future, it would be interesting to develop a logical model that captures the pragmatics of intention reconsideration by supporting control over it.

Properties

We now show that our formalization has some desirable properties. Some of these (e.g. Proposition 3a) are analogues of the AGM postulates; others (e.g. adopting logically equivalent goals has the same result, etc.) were left out for space reasons. First we show that c-goals are consistent:

Prop. 1 (Consistency) $D \models \forall s.\ \neg\text{CGoal}(\text{False}, s)$.

Thus, the agent cannot have both ϕ and $\neg\phi$ c-goals in a situation s. Even if all of the agent's p-goals become known to be impossible, the set of c-goal-accessible paths will be precisely those that starts with a K-accessible situation, and thus the agent will only choose the propositions that are known to be inevitable.

We also have the property of realism (Cohen and Levesque 1990), i.e. if an agent knows that something has become inevitable, then she has this as a c-goal:

Prop. 2 (Realism) $D \models \forall s.\ \text{KInevitable}(\phi, s) \supset \text{CGoal}(\phi, s)$.

Note that this is not necessarily true for p-goals and primary c-goals – an agent may know that something has become inevitable and not have it as her p-goal/primary c-goal, which is intuitive. While the property of realism is often criticized, one should view these inevitable goals as something that hold in the worlds that the agent intends to bring about, rather than something that the agent is actively pursuing.

A consequence of Proposition 1 and 2 is that an agent does not have a c-goal that is known to be impossible, i.e. $D \models \text{CGoal}(\phi, s) \supset \neg\text{KImpossible}(\phi, s)$.

We next discuss some properties of the framework w.r.t. goal change. Proposition 3 says that (a) an agent acquires the p-goal that ϕ at level n after she adopts it at n, and (b) that she acquires the primary c-goal that ϕ after she adopts it at some level n in s, provided that she does not have the c-goal in s that $\neg\phi$ next.

Prop. 3 (Adoption) (a) $D \models \text{PGoal}(\phi, n, do(adopt(\phi, n), s))$,
(b) $D \models \neg\text{CGoal}(\neg\exists p'.\ \text{Starts}(p, s') \land$
$\quad\quad \text{Suffix}(p', p, do(adopt(\phi, n), s')) \land \phi(p'), s)$
$\quad \supset \text{PrimCGoal}(\phi, do(adopt(\phi, n), s))$.

Also, after dropping the p-goal that ϕ at n in s, an agent does not have the p-goal (and thus the primary c-goal) that the progression of ϕ at n, i.e. ϕ', provided that ϕ' is not inevitable in $do(drop(\phi), s)$.

Prop. 4 (Drop) $D \models \text{PGoal}(\phi, n, s)$
$\quad \land [(\forall p, p'.\ \text{Starts}(p, s') \land \text{SameHist}(s', s) \land$
$\quad\quad \text{Suffix}(p', p, do(drop(\phi), s'))) \supset (\phi(p) \equiv \phi'(p'))]$
$\quad \land \neg\text{Inevitable}(\phi', do(drop(\phi), s))$
$\supset \neg\text{PGoal}(\phi', n, do(drop(\phi), s))$.

Note that, this does not hold for CGoal, as ϕ could still be a consequence of her remaining primary c-goals.

The next few properties concern the persistence of these motivational attitudes. First we have a persistence property for achievement realistic p-goals:

Prop. 5 (Persistence of Achievement RPGoals)
$D \models \text{RPGoal}(\Diamond\Phi, n, s) \land \text{Know}(\neg\Phi, s) \land \forall\psi.\ a \neq drop(\psi)$
$\quad\quad \supset \exists n'.\ \text{RPGoal}(\Diamond\Phi, n', do(a, s))$.

This says that if an agent has a realistic p-goal that $\Diamond\Phi$ in s, then she will retain this realistic p-goal after some action a has been performed in s, provided that she knows that Φ has not yet been achieved, and a is not the action of dropping a p-goal. Note that, we do not need to ensure that $\Diamond\Phi$ is consistent with higher priority active p-goals, since the SSA for G does not automatically drop such incompatible p-goals from the goal hierarchy. Also, the level n where Φ is a p-goal may change, e.g. if the action performed is an adopt action with priority higher than or equal to n.

For achievement chosen goals we have the following:

Prop. 6 (Persistence of Achievement Chosen Goals)
$D \models \text{OPGoal}(\Diamond\Phi \land \exists s'.\ \text{Starts}(s') \land \text{SameHist}(s'), n, s)$
$\quad \land \text{CGoal}(\Diamond\Phi, s) \land \text{Know}(\neg\Phi, s) \land \forall\psi.\ a \neq drop(\psi)$
$\quad \land \forall\psi, m.\ \neg(a = adopt(\psi, m) \land m \leq n)$
$\quad \land \neg\text{CGoal}(\neg\Diamond\Phi, n - 1, do(a, s))$
$\quad\quad\quad \supset \text{CGoal}(\Diamond\Phi, n, do(a, s))$.

Thus, in situation s, if an agent has the only p-goal at level n that $\Diamond\Phi$ and that the correct history of actions in s has been performed, and if $\Diamond\Phi$ is also a chosen goal in s (and thus she has the primary c-goal that $\Diamond\Phi$), then she will retain the c-goal that $\Diamond\Phi$ at level n after some action a has been performed in s, provided that: she knows that Φ has not yet been achieved, that a is not the action of dropping a p-goal, that a is not the action of adopting a p-goal at some higher priority level than n or at n, and that at level $n-1$ the agent does not have the c-goal that $\neg\Diamond\Phi$, i.e. $\Diamond\Phi$ is consistent with higher priority c-goals.

Note that, this property also follows if we replace the consequent with $\mathrm{CGoal}(\Diamond\Phi, do(a, s))$, and thus it deals with the persistence of c-goals. Note however that, it does not hold if we replace the OPGoal in the antecedent with PGoal; the reason for this is that the agent might have a p-goal at level n in s that ϕ and the c-goal in s that ϕ, but not have ϕ as a primary c-goal in s, e.g. n might be an inactive level because another p-goal at n has become impossible, and ϕ could be a c-goal in s because it is a consequence of two other primary c-goals. Thus even if $\neg\phi$ is not a c-goal after a has been performed in s, there is no guarantee that the level n will be active in $do(a, s)$ or that all the active p-goals that contributed to ϕ in s are still active.

Discussion and Future Work

While in our account chosen goals are closed under logical consequence, primary c-goals are not. Thus, our formalization of primary c-goals is related to the non-normal modal formalizations of intentions found in the literature (Konolige and Pollack 1993), and as such it does not suffer from the side-effect problem (Cohen and Levesque 1990).

Our framework can be extended to model subgoal adoption and the dependencies between goals and the subgoals and plans adopted to achieve them. The later is important since subgoals and plans adopted to bring about a goal should be dropped when the parent goal becomes impossible, is achieved, or is dropped. One way of handling this is to ensure that the adoption of a subgoal ψ w.r.t. a parent goal ϕ adds a new p-goal that contains *both this subgoal and this parent goal*, i.e. $\psi \wedge \phi$. This ensures that when the parent goal is dropped, the subgoal is also dropped, since when we drop the parent goal ϕ, we drop all the p-goals at all G-accessibility levels that imply ϕ including $\psi \wedge \phi$.

Also, since we are using the situation calculus, we can easily represent procedural goals/plans, e.g. the goal to do a_1 and then a_2 can be written as: $\mathrm{PGoal}(\mathrm{Starts}(p, s_1) \wedge \mathrm{OnPath}(p, s) \wedge s = do(a_2, do(a_1, s_1)), 0, S_0)$. Golog (Reiter 2001) can be used to represent complex plans/programs. So we can model the adoption of plans as subgoals.

Recently, there have been a few proposals that deal with goal change. Shapiro *et al.* (2007) present a situation calculus based framework where an agent adopts a goal when she is requested to do so, and remains committed to this goal unless the requester cancels this request; a goal is retained even if the agent learns that it has become impossible, and in this case the agent's goals become inconsistent. Shapiro and Brewka (2007) modify this framework to ensure that goals are dropped when they are believed to be impossible or when they are achieved. Their account is similar to ours in the sense that they also assume a priority ordering over the set of (in their case, requested) goals, and in every situation they compute chosen goals by computing a maximal consistent goal set that is also compatible with the agent's beliefs. However, their model has some unintuitive properties: the agent's chosen set of goals in $do(a, s)$ may be quite different from her goals in s, although a did not make any of her goals in s impossible or inconsistent with higher priority goals, because inconsistencies between goals at the same priority level are resolved differently (this can happen because goals are only partially ordered). Also, we provide a more expressive formalization of prioritized goals – we model goals using infinite paths, and thus can model many types of goals that they cannot. Finally they model prioritized goals by treating the agent's p-goals as an arbitrary set of temporal formulas, and then defining the set of c-goals as a subset of the p-goals. But our possible world semantics has some advantages over this: it clearly defines when goals are consistent with each other and with what is known. One can easily specify how goals change when an action a occurs, e.g. the goal to do a next and then do b becomes the goal to do b next, the goal that $\Diamond\Phi \vee \Diamond\Psi$ becomes the goal that $\Diamond\Psi$ if a makes achieving Φ impossible, etc.

Most approaches to agent programming languages with declarative goals are not based on a formal theory of agency, and to the best of our knowledge none deals with temporally extended goals or maintain the consistency of (chosen) goals.

One limitation of our account is that one could argue that our agent wastes resources trying to optimize her c-goals at every step. In the future, we would like to develop an account where the agent is strongly committed to her chosen goals, and where the filter override mechanism is only triggered under specific conditions.

References

M. Bratman. *Intentions, Plans, and Practical Reason*. Harvard University Press, Cambridge, 1987.

P. Cohen and H. Levesque. Intention is Choice with Commitment. *Artificial Intelligence*, 42(2–3):213–361, 1990.

K. Konolige and M. Pollack. A Representationalist Theory of Intention. In *Thirteenth Intl. J. Conf. on Artificial Intelligence (IJCAI-93)*, pp. 390–395, Chambéry, France, 1993.

H. Levesque, F. Pirri, and R. Reiter. Foundations for a Calculus of Situations. *Electronic Transactions of AI (ETAI)*, 2(3–4):159–178, 1998.

A. Rao and M. Georgeff. Modeling Rational Agents with a BDI-Architecture. In R. Fikes and E. Sandewall, editors, *Second Intl. Conf. on Principles of Knowledge Rep. and Reasoning*, pp. 473–484, 1991.

R. Reiter. *Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, Cambridge, MA, 2001.

R. Scherl and H. Levesque. Knowledge, Action, and the Frame Problem. *Artificial Intelligence*, 144(1–2), 2003.

S. Shapiro and G. Brewka. Dynamic Interactions Between Goals and Beliefs. In *Twentieth Intl. J. Conf. on Artificial Intelligence (IJCAI-07)*, pp. 2625–2630, India, 2007.

S. Shapiro, Y. Lespérance, and H. Levesque. Goals and Rational Action in the Situation Calculus - A Preliminary Report. In *Working Notes of the AAAI Fall Symp. on Rational Agency*, pp. 117–122, 1995.

S. Shapiro, Y. Lespérance, and H. Levesque. Goal Change in the Situation Calculus. *J. of Logic and Computation*, 17(5):983–1018, 2007.

A Semantical Account of Progression in the Presence of Defaults

Gerhard Lakemeyer
Dept. of Computer Science
RWTH Aachen
52056 Aachen
Germany
gerhard@cs.rwth-aachen.de

Hector J. Levesque
Dept. of Computer Science
University of Toronto
Toronto, Ontario
Canada M5S 3A6
hector@cs.toronto.edu

Abstract

In previous work, we proposed a modal fragment of the situation calculus called $\mathcal{ES}$, which fully captures Reiter's basic action theories. $\mathcal{ES}$ also has epistemic features, including only-knowing, which refers to all that an agent knows in the sense of having a knowledge base. While our model of only-knowing has appealing properties in the static case, it appears to be problematic when actions come into play. First of all, its utility seems to be restricted to an agent's initial knowledge base. Second, while it has been shown that only-knowing correctly captures default inferences, this was only in the static case, and undesirable properties appear to arise in the presence of actions. In this paper, we remedy both of these shortcomings and propose a new dynamic semantics of only-knowing, which is closely related to Lin and Reiter's notion of progression when actions are performed and where defaults behave properly.

Introduction

In previous work, Lakemeyer and Levesque (2004; 2005) proposed a modal fragment of the situation calculus called $\mathcal{ES}$, which fully captures Reiter's basic action theories and regression-based reasoning, including reasoning about knowledge. So, for example, the language allows us to formulate Reiter-style successor state axioms such as this one:

$$\begin{aligned}\forall a, x.\Box([a]\mathit{Broken}(x) \;\equiv\; & \\ &(a = \mathit{drop}(x) \wedge \mathit{Fragile}(x)) \vee \\ &(\mathit{Broken}(x) \wedge a \neq \mathit{repair}(x)))\end{aligned}$$

In English: after any sequence of actions ($\Box$), an object x will be broken after doing action a ($[a]\mathit{Broken}(x)$) iff a is the dropping of x when x is fragile, or x was already broken and a is not the action of repairing it. Here we assume that *Fragile* is a predicate which is not affected by any action so that its successor state axiom would be

$$\forall a, x.\Box([a]\mathit{Fragile}(x) \;\equiv\; \mathit{Fragile}(x)).$$

Let us call the conjunction of these two axioms SSA_{BF}. In addition to action and change, the language $\mathcal{ES}$ also addresses what an agent knows and only-knows. The latter is intended to capture all an agent knows in the sense of having a knowledge base. For illustration, consider the following sentence, which is logically valid in $\mathcal{ES}$:

$$\begin{aligned}&\boldsymbol{O}(\mathit{Fragile}(o) \wedge \neg\mathit{Broken}(o) \wedge \mathit{SSA}_{BF}) \;\supset \\ &\qquad [\mathit{drop}(o)]\,(\boldsymbol{K}(\mathit{Broken}(o)) \wedge \neg\boldsymbol{K}(\mathit{Glass}(o)))\,.\end{aligned}$$

In English: if all the agent knows is that *o* is fragile and not broken and that the successor state axioms for *Broken* and *Fragile* hold, then after dropping *o*, the agent knows that *o* is broken, but does not know that *o* is made of glass.

Let us now consider what the agent should only-know after the drop action has occurred. Intuitively, the agent's knowledge should change in that it now believes that *o* is broken, with everything else remaining the same. Formally,

$$[\mathit{drop}(o)]\,\boldsymbol{O}(\mathit{Fragile}(o) \wedge \mathit{Broken}(o) \wedge \mathit{SSA}_{BF}).$$

In fact this view corresponds essentially to what Lin and Reiter (LR) [1997] call the *progression* of a database wrt an action. It turns out, however, that the semantics of only-knowing as proposed in (Lakemeyer and Levesque 2004) differs from this in that the last formula above is *not* entailed. The reason is that their version, unlike progression, does not forget what was true initially (like whether or not *o* was already broken), and so more ends up being known.

The LR notion of progression allows for efficient implementations under certain restrictions (Lin and Reiter 1997; Liu and Levesque 2005; Vassos and Levesque 2007), and being able to forget the past seems essential for this. Hence the previous semantics of only-knowing may not be very useful, except perhaps in the initial state. In this paper, we present a new semantics of only-knowing which avoids this pitfall and is fully compatible with LR's idea of progression.

Levesque (1990) showed that only-knowing in the static case also accounts for default reasoning in the sense of autoepistemic logic (Moore 1985). For example, the default that objects are fragile unless known otherwise can be written as

$$\forall x \neg\boldsymbol{K}\neg\mathit{Fragile}(x) \supset \mathit{Fragile}(x).$$

If the agent uses this default instead of the fact that *o* is fragile then it would still conclude, this time by default, that *o* is fragile and hence believe that it is broken after dropping it. But suppose that *o* is actually *not* fragile. What should the agent believe after *sensing* the status of *o*'s fragility? Clearly, it should then believe that *o* is indeed not fragile and it should not believe that dropping *o* will break it. That is, the default should no longer apply. Unfortunately, the previous definition of only-knowing does not do this. The problem, roughly, is that the initial default conclusion that *o* is fragile cannot be distinguished from a hard fact. Subsequently sensing the opposite then leads to an inconsistency.

In this paper we will fix this problem by proposing a semantics which separates conclusions based on facts from those based on defaults. To this end, we will distinguish between what is known for sure (using the modality $\boldsymbol{K}$) and what is believed after applying defaults (using another modality $\boldsymbol{B}$). In fact, defaults themselves will be formulated using $\boldsymbol{B}$ instead of $\boldsymbol{K}$. All this will be integrated with progression in the sense that defaults will be applied to the progressed knowledge base.

For space reasons, the paper, which also appears in (Lakemeyer and Levesque 2009a) in almost identical form, contains no proofs. These and a comparison between the old and new semantics of only-knowing and between our notion of progression and that of LR can be found in (Lakemeyer and Levesque 2009b).

The rest of the paper is organized as follows. In the next section, we introduce the logic $\mathcal{ES}_{\text{O}}$, which is like the old $\mathcal{ES}$ except for the new semantics of only-knowing and defaults. This semantics agrees with the previous one in the static case. After that, we consider only-knowing in the context of basic action theories. In particular, we show that what is only-known after an action extends LR's original idea of progression, and how reasoning about defaults fits into the picture. We then address related work and conclude.

The Logic $\mathcal{ES}_{\text{O}}$

The language is a second-order modal dialect with equality and sorts of type object and action. Before presenting the formal details, here are the main features:

- *rigid terms*: The ground terms of the language are taken to be isomorphic to the domain of discourse. This allows first-order quantification to be understood substitutionally. Equality can also be given a very simple treatment: two ground terms are equal only if they are identical.
- *knowledge and truth*: The language includes modal operators $\boldsymbol{K}$ and $\boldsymbol{B}$ for knowledge and belief. The $\boldsymbol{K}$ operator allows us to distinguish between sentences that are true and sentences that are known (by some implicit agent). The $\boldsymbol{B}$ operator allows an agent to have false beliefs about its world or how its world changes. For example, we can model situations where an object is not fragile but the agent does not know it, yet may believe that it is fragile by default.
- *sensing*: The connection between knowledge and truth is made with sensing. Every action is assumed to have a binary sensing result and after performing the action, the agent learns that the action was possible (as indicated by the *Poss* predicate) and whether the sensing result for the action was 1 or 0 (as indicated by the *SF* predicate).[1] Just as an action theory may contain precondition axioms characterizing the conditions under which *Poss* holds, it can contain axioms characterizing the conditions under which *SF* holds.

The language

The symbols of $\mathcal{ES}_{\text{O}}$ consist of first-order variables, second-order predicate variables of every arity, rigid functions of every arity, fluent predicate symbols of every arity, as well as these connectives and other symbols: $=$, $\wedge$, $\neg$, $\forall$, $\boldsymbol{K}$, $\boldsymbol{B}$, $\boldsymbol{O}$, $\boldsymbol{\Omega}$, $\Box$, round and square parentheses, period, comma. We assume two special fluent predicates *Poss* and *SF* (for sensing). $\boldsymbol{K}$, $\boldsymbol{B}$, $\boldsymbol{O}$, and $\boldsymbol{\Omega}$ are called epistemic operators.

The *terms* of the language are formed in the usual way from first-order variable and rigid functions.

We let $\mathcal{R}$ denote the set of all rigid terms (here, all ground terms). For simplicity, instead of having variables of the *action* sort distinct from those of the *object* sort as in the situation calculus, we lump both of these together and allow ourselves to use any term as an action or as an object.[2]
The *well-formed formulas* of the language form the least set such that

1. If $t_1, \ldots, t_k$ are terms, F is a k-ary predicate symbol, and V is a k-ary second-order variable, then $F(t_1, \ldots, t_k)$ and $V(t_1, \ldots, t_k)$ are (atomic) formulas;
2. If t_1 and t_2 are terms, then $(t_1 = t_2)$ is a formula;
3. If α and β are formulas, v is a first-order variable, V is a second-order variable, and t is a term, then the following are also formulas: $(\alpha \wedge \beta)$, $\neg\alpha$, $\forall v.\, \alpha$, $\forall V.\, \alpha$, $[t]\alpha$, $\Box\alpha$, $\boldsymbol{K}\alpha$, $\boldsymbol{B}\alpha$, $\boldsymbol{O}\alpha$, and $\boldsymbol{\Omega}\alpha$, where the formulas following $\boldsymbol{O}$ and $\boldsymbol{\Omega}$ are restricted further below.

We read $[t]\alpha$ as "α holds after action t", and $\Box\alpha$ as "α holds after any sequence of actions," and $\boldsymbol{K}\alpha$ ($\boldsymbol{B}\alpha$) as "the agent knows (believes) α." $\boldsymbol{O}\alpha$ may be read as "the agent only-knows α" and is intended to capture all the agent knows about what the world is like now and how it evolves as a result of actions. Here no defaults are taken into account, just facts which, as we will see later, come in the form of a basic action theory similar to those proposed by Reiter (2001a). Therefore, we restrict $\boldsymbol{O}$ to apply to so-called *objective formulas* only, which are those mentioning no epistemic operators. Finally, $\boldsymbol{\Omega}\alpha$ is meant to capture all and only the defaults believed by the agent. For that, α is restricted to what we call *static belief formulas*, which mention neither $\Box$ nor $[t]$ nor any epistemic operator except $\boldsymbol{B}$.

As usual, we treat $(\alpha \vee \beta)$, $(\alpha \supset \beta)$, $(\alpha \equiv \beta)$, $\exists v.\, \alpha$, and $\exists V.\, \alpha$ as abbreviations. We use α^x_t to mean formula α with all free occurrences of variable x replaced by term t. We call a formula without free variables a *sentence*.

We will also sometimes refer to *static objective formulas*, which are the objective formulas among the static belief formulas, and *fluent formulas*, which are formulas with no $\boldsymbol{K}$, $\boldsymbol{O}$, $\boldsymbol{B}$, $\boldsymbol{\Omega}$, $\Box$, $[t]$, *Poss*, or *SF*.[3]

[1] For convenience, we assume that every action returns a (perhaps trivial) sensing result. Here, we restrict ourselves to binary values. See (Scherl and Levesque 2003) for how to handle arbitrary sensing results.

[2] Equivalently, the version in this paper can be thought of as having action terms but no object terms.

[3] In the situation calculus, these correspond to formulas that are uniform in some situation term.

The semantics

The main purpose of the semantics we are about to present is to be precise about how we handle fluents, which may vary as the result of actions and whose values may be unknown. Intuitively, to determine whether or not a sentence α is true after a sequence of actions z has been performed, we need to specify two things: a world w and an epistemic state e. A world determines truth values for the ground atoms after any sequence of actions. An epistemic state is defined by a set of worlds, as in possible-world semantics.

More precisely, let $\mathcal{Z}$ be the set of all finite sequences of elements of $\mathcal{R}$ including $\langle\,\rangle$, the empty sequence. $\mathcal{Z}$ should be understood as the set of all finite sequences of actions. Then

- a world $w \in W$ is any function from $\mathcal{G}$ (the set of ground atoms) and $\mathcal{Z}$ to $\{0, 1\}$.
- an epistemic state $e \subseteq W$ is any set of worlds.

To interpret formulas with free variables, we proceed as follows. First-order variables are handled substitutionally using the rigid terms $\mathcal{R}$. To handle the quantification over second-order variables, we use second-order *variable maps* defined as follows:

> The *second-order ground atoms* are formulas of the form $V(t_1, \ldots, t_k)$ where V is a second-order variable and all of the t_i are in $\mathcal{R}$. A *variable map* u is a function from second-order ground atoms to $\{0, 1\}$.

Let u and u' be variable maps, and let V be a second-order variable; we write $u' \sim_V u$ to mean that u and u' agree except perhaps on the assignments involving V.

Finally, to interpret what is known after a sequence of actions has taken place, we define $w' \simeq_z w$ (read: w' agrees with w on the sensing throughout action sequence z) inductively by the following:

1. $w' \simeq_{\langle\,\rangle} w$ for all w';
2. $w' \simeq_{z\cdot t} w$ iff $w' \simeq_z w$,
   $w'[\mathit{Poss}(t), z] = 1$ and $w'[\mathit{SF}(t), z] = w[\mathit{SF}(t), z]$.

Note that $\simeq_z$ is not quite an equivalence relation because of the use of *Poss* here. This is because we are insisting that the agent comes to believe that *Poss* was true after performing an action, even in those "non-legal" situations where the action was not possible in reality.[4]

Putting all these together, we now turn to the semantic definitions for sentences of $\mathcal{ES}_{\mathrm{O}}$. Given an epistemic state $e \subseteq W$, a world $w \in W$, an action sequence $z \in \mathcal{Z}$, and a second-order variable map u, we have:

1. $e, w, z, u \models F(t_1, \ldots, t_k)$ iff $w[F(t_1, \ldots, t_k), z] = 1$;
2. $e, w, z, u \models V(t_1, \ldots, t_k)$ iff $u[V(t_1, \ldots, t_k)] = 1$;
3. $e, w, z, u \models (t_1 = t_2)$ iff t_1 and t_2 are identical;
4. $e, w, z, u \models [t]\alpha$ iff $e, w, z \cdot t, u \models \alpha$;
5. $e, w, z, u \models (\alpha \wedge \beta)$ iff
   $e, w, z, u \models \alpha$ and $e, w, z, u \models \beta$;
6. $e, w, z, u \models \neg\alpha$ iff $e, w, z, u \not\models \alpha$;
7. $e, w, z, u \models \forall x.\, \alpha$ iff $e, w, z, u \models \alpha^x_t$, for all $t \in \mathcal{R}$;
8. $e, w, z, u \models \forall V.\, \alpha$ iff
   $e, w, z, u' \models \alpha$, for all $u' \sim_V u$;
9. $e, w, z, u \models \Box\alpha$ iff $e, w, z \cdot z', u \models \alpha$, for all $z' \in \mathcal{Z}$;

To define the meaning of the epistemic operators, we need the following definition:

Definition 1 *Let w be a world and e a set of worlds, and z a sequence of actions. Then*

1. *w_z is a world such that $w_z[p, z'] = w[p, z \cdot z']$ for all ground atoms p and action sequences z';*
2. *$e^w_z = \{w'_z \mid w' \in e$ and $w' \simeq_z w\}$.*

Note that w_z is exactly like w after the actions z have occurred. So in a sense, w_z can be thought of as the progression of w wrt z. e^w_z then contains all those worlds of e which are progressed wrt z and which are compatible with (the real) world w in terms of the sensing results and where all the actions in z are executable. Note that when z is empty, $e^w_z = e$.

10. $e, w, z, u \models \boldsymbol{K}\alpha$ iff
    for all $w' \in e^w_z$, $e^w_z, w', \langle\rangle, u \models \alpha$;
11. $e, w, z, u \models \boldsymbol{O}\alpha$ iff
    for all w', $w' \in e^w_z$ iff $e^w_z, w', \langle\rangle, u \models \alpha$.

In other words, knowing α in e and w after actions z means that α is true in all the progressed worlds of e which are compatible with w. $\boldsymbol{O}\alpha$ is quite similar except for the "iff," whose effect is that e^w_z must contain every world which satisfies α.

$\boldsymbol{B}$ and $\boldsymbol{\Omega}$ are meant to capture what the agent believes in addition by applying defaults. Having more beliefs (as a result of defaults) is modeled by considering a subset of the worlds in e^w_z. For that purpose, we introduce a function δ which maps each set of worlds into a subset. In particular, we require that $\delta(e^w_z) \subseteq e^w_z$. As δ is now part of the model (just like w and e) we add it to the L.H.S. of the satisfaction relation with the understanding that the previous rules are retrofitted with δ as well. Then we have:

12. $e, w, z, u, \delta \models \boldsymbol{B}\alpha$ iff
    for all $w' \in \delta(e^w_z)$, $e^w_z, w', \langle\rangle, u, \delta \models \alpha$;
13. $e, w, z, u, \delta \models \boldsymbol{\Omega}\alpha$ iff
    for all $w' \in e^w_z$, $w' \in \delta(e^w_z)$ iff $e^w_z, w', \langle\rangle, u, \delta \models \alpha$.

Note that the only difference between $\boldsymbol{K}$ and $\boldsymbol{B}$ is that the latter considers $\delta(e^w_z)$ instead of e^w_z. Likewise, the definition of $\boldsymbol{\Omega}$ is similar to that of $\boldsymbol{O}$. The role of $\boldsymbol{\Omega}$ is to constrain δ to produce a special subset of e^w_z. Roughly, the effect of the definition of $\boldsymbol{\Omega}\alpha$ is that one starts with whatever facts are believed (represented by e^w_z) and then settles on a largest subset of e^w_z such that α (representing the defaults) is also believed.

We say that a sentence in $\mathcal{ES}_{\mathrm{O}}$ is true at a given e, w, and δ (written $e, w, \delta \models \alpha$) if $e, w, \langle\,\rangle, u, \delta \models \alpha$ for any second-order variable map u. If Σ is a set of sentences and α is a sentence, we write $\Sigma \models \alpha$ (read: Σ logically entails α) to mean that for every e, w, and δ, if $e, w, \delta \models \alpha'$ for every

[4] An alternate account that would state that the agent learns the true value of *Poss* (analogous to *SF*) is a bit more cumbersome, but would allow $\simeq_z$ to be a full equivalence relation.

$\alpha' \in \Sigma$, then $e, w, \delta \models \alpha$. Finally, we write $\models \alpha$ (read: α is valid) to mean $\{\} \models \alpha$.

For reasons of space we cannot go into details about the general logical properties of the epistemic operators. To demonstrate that the operators are well-behaved, we only list some properties, which all have simple semantic proofs:
$\models \Box(\boldsymbol{K}\alpha \supset \boldsymbol{B}\alpha)$
$\models \Box(\boldsymbol{O}\alpha \supset \boldsymbol{K}\alpha)$
$\models \Box(\boldsymbol{\Omega}\alpha \supset \boldsymbol{B}\alpha)$

Moreover, $\boldsymbol{K}$ and $\boldsymbol{B}$ satisfy the usual $K45$ axioms of modal logic (Hughes and Cresswell 1968) and they are mutually introspective, e.g. $\models \Box(\boldsymbol{B}\alpha \supset \boldsymbol{KB}\alpha)$.

The Semantics of Progression and Defaults

Basic action theories

Let us now consider the equivalent of basic action theories of the situation calculus. Since in our logic there is no explicit notion of situations, our basic action theories do not require foundational axioms like Reiter's (2001a) second-order induction axiom for situations. The treatment of defaults is deferred to Section .

Definition 2 *Given a set of fluents $\mathcal{F}$, a set $\Sigma \subseteq \mathcal{ES}_0$ of sentences is called a basic action theory over $\mathcal{F}$ iff* $\Sigma = \Sigma_0 \cup \Sigma_{\text{pre}} \cup \Sigma_{\text{post}} \cup \Sigma_{\text{sense}}$ *where*

1. *Σ_0 is any set of fluent sentences;*
2. *Σ_{pre} is a singleton sentence of the form $\Box Poss(a) \equiv \pi$, where π is a fluent formula;*[5]
3. *Σ_{post} is a set of sentences of the form $\Box[a]F(\vec{v}) \equiv \gamma_F$, one for each relational fluent F in $\mathcal{F}$, respectively, and where the γ_F are fluent formulas.*[6]
4. *Σ_{sense} is a sentence exactly parallel to the one for Poss of the form $\Box SF(a) \equiv \varphi$, where φ is a fluent formula.*

The idea here is that Σ_0 expresses what is true initially (in the initial situation), Σ_{pre} is one large precondition axiom, and Σ_{post} is a set of successor state axioms, one per fluent in $\mathcal{F}$, which incorporate the solution to the frame problem proposed by Reiter (1991). Σ_{sense} characterizes the sensing results of actions. For actions like *drop*(o), which do not return any useful sensing information, *SF* can be defined to be vacuously true (see below for an example).

We will usually require that Σ_{pre}, Σ_{post}, and Σ_{sense} be first-order. However, Σ_0 may contain second-order sentences. As we will see, this is inescapable if we want to capture progression correctly. In the following, we assume that Σ (and hence $\mathcal{F}$) is finite and we will freely use Σ or its subsets as part of sentences with the understanding that we mean the conjunction of the sentences contained in the set.

Progression = Only-knowing after an action

Let us now turn to the first main result of this paper. The question we want to answer is this: suppose an agent is given a basic action theory as its initial knowledge base; how do we characterize the agent's knowledge after an action is performed? As hinted in the introduction, only-knowing will give us the answer.

In the following, for a given basic action theory Σ, we sometimes write ϕ for Σ_0 and $\Box\beta$ for the rest of the action theory $\Sigma_{\text{pre}} \cup \Sigma_{\text{post}} \cup \Sigma_{\text{sense}}$. We assume that π and φ refer to the right-hand sides of the definitions of *Poss* and *SF* in Σ, and γ_F is the right-hand side of the successor state axiom for fluent F. Also, let $\vec{F}$ consist of all the fluent predicate symbols in Σ, and let $\vec{P}$ be corresponding second-order variables, where each P_i has the same arity as F_i. Then $\alpha^{\vec{F}}_{\vec{P}}$ denotes the formula α with every occurrence of F_i replaced by P_i.

The following result characterizes in general terms all that is known after performing an action:

Theorem 1 *Let $\Sigma = \phi \wedge \Box\beta$ be a basic action theory and t an action term. Then*

$$\begin{aligned}\models \boldsymbol{O}(\phi \wedge \Box\beta) \supset \\ &(SF(t) \supset [t]\boldsymbol{O}(\Psi \wedge \Box\beta)) \wedge \\ &(\neg SF(t) \supset [t]\boldsymbol{O}(\Psi' \wedge \Box\beta)),\end{aligned}$$

where $\Psi = \exists\vec{P}[(\phi \wedge \pi^a_t \wedge \varphi^a_t)^{\vec{F}}_{\vec{P}} \wedge \bigwedge \forall\vec{x}.F(\vec{x}) \equiv \gamma_F{}^a_t{}^{\vec{F}}_{\vec{P}}]$ *and*

$$\Psi' = \exists\vec{P}[(\phi \wedge \pi^a_t \wedge \neg\varphi^a_t)^{\vec{F}}_{\vec{P}} \wedge \bigwedge \forall\vec{x}.F(\vec{x}) \equiv \gamma_F{}^a_t{}^{\vec{F}}_{\vec{P}}].$$

What the theorem says is that if all the agent knows initially is a basic action theory, then after doing action t all the agent knows is another basic action theory, where the dynamic part $(\Box\beta)$ remains the same and the initial database ϕ is replaced by Ψ or Ψ', depending on the outcome of the sensing. Note that the two sentences differ only in one place, φ^a_t vs. $\neg\varphi^a_t$. Roughly, Ψ and Ψ' specify how the truth value of each fluent F in $\mathcal{F}$ is determined by what was true previously (ϕ), taking into account that the action was possible (π^a_t) and that the sensing result was either true (φ^a_t) or false ($\neg\varphi^a_t$). Since after performing an action, the agent again only-knows a basic action theory, we can take this as its new initial theory and the process can iterate. We remark that our notion of progression is very closely related to progression as defined by (Lin and Reiter 1997), but extends it to handle sensing actions. Note that, while Lin and Reiter need to include the unique names axioms for actions in the progression, we do not, as these are built into the logic.

We mentioned above that after an action, the resulting knowledge base can be taken as the new initial knowledge base, and the progression can iterate. The following theorem shows that this view is justified in that the entailments about the future remain the same when we substitute what is known about the world initially by its progression. Here we only consider the case where *SF*(t) is true.

Theorem 2 $\models \boldsymbol{O}(\phi \wedge \Box\beta) \wedge SF(t) \supset [t]\boldsymbol{K}(\alpha)$ *iff*
$\models \boldsymbol{O}(\Psi \wedge \Box\beta) \supset \boldsymbol{K}(\alpha)$.

In English (roughly): It follows from your initial knowledge base that you will know α after doing action t iff knowing α follows from your progressed knowledge base.

[5] We assume that all free variables are implicitly universally quantified and that $\Box$ has lower syntactic precedence than the logical connectives, so that $\Box Poss(a) \equiv \pi$ stands for the sentence $\forall a.\Box(Poss(a) \equiv \pi)$.

[6] The $[t]$ construct has higher precedence than the logical connectives. So $\Box[a]F(\vec{x}) \equiv \gamma_F$ abbreviates the sentence $\forall a, \vec{x}.\Box([a]F(\vec{x}) \equiv \gamma_F)$.

Defaults for basic action theories

Here we restrict ourselves to static defaults like "birds normally fly." In an autoepistemic setting (Moore 1985; Levesque 1990), these have the following form:

$$\forall\vec{x}.\boldsymbol{B}\alpha \wedge \neg\boldsymbol{B}\neg\beta \supset \gamma,$$

which can be read as "if α is believed and β is consistent with what is believed then assume γ." Here the assumption is that α, β, and γ are static objective formulas.

Let Σ_{def} be the conjunction of all defaults of the above form held by an agent. For a given basic action theory Σ, as defined in Section , the idea is to apply the same defaults to what is known about the current situation after any number of actions have occurred, that is, for the purpose of default reasoning, we assume that $\Box\Omega\Sigma_{\text{def}}$ holds. The following theorem relates what is then believed after one action has occurred (where *SF* returns true) with stable expansions (Moore 1985).[7]

Theorem 3 *Let t be a ground action and $\Sigma = \phi \wedge \Box\beta$ a basic action theory such that $\models \boldsymbol{O}\Sigma \wedge \mathit{SF}(t) \supset [t]\boldsymbol{O}(\psi \wedge \Box\beta)$ and ψ is first order. Then for any static belief sentence γ,*

$$\models \boldsymbol{O}\Sigma \wedge \mathit{SF}(t) \wedge \Box\Omega\Sigma_{\text{def}} \supset [t]\boldsymbol{B}\gamma \quad \textit{iff}$$
$$\gamma \textit{ is in every stable expansion of } \psi \wedge \Sigma_{\text{def}}.$$

An example

To illustrate progression, let us consider the example of the introduction with two fluents *Broken* and *Fragile*, actions $\mathit{drop}(x)$, $\mathit{repair}(x)$, and $\mathit{senseF}(x)$ (for sensing whether x is fragile). First, we let the basic action theory Σ consist of the following axioms:

- $\Sigma_0 = \{\mathit{Fragile}(o), \neg\mathit{Broken}(o)\}$;
- $\Sigma_{\text{pre}} = \{\Box\mathit{Poss}(a) \equiv \mathit{true}\}$ (for simplicity);
- $\Sigma_{\text{post}} = \{\mathit{SSA}_{BF}\}$ (from the introduction);
- $\Sigma_{\text{sense}} = \{\Box\mathit{SF}(a) \equiv \exists x.a = \mathit{drop}(x) \wedge \mathit{true} \vee a = \mathit{repair}(x) \wedge \mathit{true} \ \vee\ a = \mathit{senseF}(x) \wedge \mathit{Fragile}(x)\}$.

As before, let $\Box\beta$ be $\Sigma_{\text{pre}} \cup \Sigma_{\text{post}} \cup \Sigma_{\text{sense}}$. Then we have

$$\models \Sigma \wedge \boldsymbol{O}(\Sigma_0 \wedge \Box\beta) \supset [\mathit{drop}(o)]\boldsymbol{O}(\Psi \wedge \Box\beta),$$

where
$$\begin{aligned}\Psi = \exists P, P'.[&\neg P(o) \wedge P'(o) \wedge\\ &\exists x.\mathit{drop}(o) = \mathit{drop}(x) \wedge \mathit{true} \vee\\ &\quad \mathit{drop}(o) = \mathit{repair}(x) \wedge \mathit{true} \vee\\ &\quad \mathit{drop}(o) = \mathit{senseF}(x) \wedge P'(x) \wedge\\ &\forall x. \mathit{Broken}(x) \equiv \mathit{drop}(o) = \mathit{drop}(x) \wedge P'(x) \vee\\ &\quad P(x) \wedge \mathit{drop}(o) \neq \mathit{repair}(x) \wedge\\ &\forall x. \mathit{Fragile}(x) \equiv P'(x)].\end{aligned}$$

Using the fact that all actions are distinct, it is not difficult to see that Ψ can be simplified to

$$(\mathit{Fragile}(o) \wedge \mathit{Broken}(o)).$$

In other words, after dropping *o*, the agent's knowledge base is as before, except that *o* is now known to be broken.

To see how defaults work, we now let Σ be as before except that $\Sigma_0 = \{\neg\mathit{Broken}(o)\}$ and let $\Sigma' = \Sigma \cup \{\neg\mathit{Fragile}(o)\}$. Let $\Sigma_{\text{def}} = \{\forall x.\neg\boldsymbol{B}\neg\mathit{Fragile}(x) \supset \mathit{Fragile}(x)\}$. Then the following are logical consequences of

$$\Sigma' \wedge \boldsymbol{O}(\Sigma_0 \wedge \Box\beta) \wedge \Box\Omega\Sigma_{\text{def}} \ :$$

1. $\boldsymbol{B}\mathit{Fragile}(o)$;
2. $[\mathit{drop}(o)]\boldsymbol{B}\mathit{Broken}(o)$;
3. $[\mathit{senseF}(o)]\boldsymbol{K}\neg\mathit{Fragile}(o)$;
4. $[\mathit{senseF}(o)][\mathit{drop}(o)]\boldsymbol{K}\neg\mathit{Broken}(o)$.

(1) holds because of the default, since o's non-fragility is not yet known. Notice, in particular, the role of $\Omega\Sigma_{\text{def}}$: while the semantics of $\mathcal{ES}_0$ puts no restrictions on δ other than $\delta(e) \subseteq e$,[8] it is $\Omega\Sigma_{\text{def}}$ which forces $\delta(e)$ to be the largest subset of e which is compatible with the default, that is, δ selects only worlds from e where o is fragile. (2) holds because the default also applies after $\mathit{drop}(o)$. In particular, Theorem 3 applies as $[\mathit{drop}(o)]\boldsymbol{O}(\mathit{Broken}(o) \equiv \mathit{Fragile}(o) \wedge \Box\beta)$ follows as well. Finally, in (3) and (4) the agent has found out that o is not fragile, blocking the default since $\models \Box(\boldsymbol{K}\alpha \supset \boldsymbol{B}\alpha)$.

As one of the reviewers remarked, from a commonsense point of view, it is also or perhaps more plausible to have a sensing action for broken instead of fragility. In other words, after dropping an object one would sense whether it is broken, and if not conclude that it must not be fragile. This can be modeled in our framework as well. In particular, in all situations which are compatible with sensing that the object o is not broken after dropping it the fluent $\mathit{Fragile}(o)$ is false.

Related Work

While the situation calculus has received a lot of attention in the reasoning about action community, there are, of course, a number of alternative formalisms, including close relatives like the fluent calculus (Thielscher 1999) and more distant cousins such as (Kowalski and Sergot 1986; Gelfond and Lifschitz 1993).

While $\mathcal{ES}_0$ is intended to capture a fragment of the situation calculus, it is also related to the work formalizing action and change in the framework of dynamic logic (Harel 1984). Examples are (De Giacomo and Lenzerini 1995) and later (Herzig *et al* 2000), who also deal with belief. While these approaches remain propositional, there are also first-order treatments such as (Demolombe 2003; Demolombe, Herzig, and Varzinczak 2003; Blackburn *et al* 2001), which, like $\mathcal{ES}_0$, are inspired by the desire to capture fragments of the situation calculus in modal logic. Demolombe (2003) even considers a form of only-knowing, which is related to the version of only-knowing in (Lakemeyer and Levesque 2004), which in turn derives from the logic $\mathcal{OL}$ (Levesque and Lakemeyer 2001).

The idea of progression is not new and lies at the heart of most planning systems, starting with STRIPS (Fikes and Nilsson 1971), but also in implemented agent programming languages like 3APL (Hindriks *et al* 1999). Lin and Reiter (1997) so far gave the most general account. Restricted forms of LR-progression, which are first-order definable, are

[7] Roughly, E is a stable expansion of α iff for all γ, $\gamma \in E$ iff γ is a first-order consequence of $\{\alpha\} \cup \{\boldsymbol{B}\beta \mid \beta \in E\} \cup \{\neg\boldsymbol{B}\beta \mid \beta \notin E\}$.

[8] Here e is the (unique) set of worlds which satisfies $\boldsymbol{O}(\Sigma_0 \wedge \Box\beta)$.

discussed in (Lin and Reiter 1997; Liu and Levesque 2005; Claßen *et al* 2007; Vassos and Levesque 2007).

Default reasoning has been applied to actions mostly to solve the frame problem (Shanahan 1993). Here, however, we use Reiter's monotonic solution to the frame problem (Reiter 1991) and we are concerned with the static "Tweety-flies" variety of defaults. Kakas et al. (2008) recently made a proposal that deals with these in the presence of actions, but only in a propositional setting of a language related to $\mathcal{A}$ (Gelfond and Lifschitz 1993).

Conclusion

The paper introduced a new semantics for the concept of only-knowing within a modal fragment of the situation calculus. In particular, we showed that, provided an agent starts with a basic action theory as its initial knowledge base, then all the agent knows after an action is again a basic action theory. The result is closely related to Lin and Reiter's notion of progression and generalizes it to allow for actions which return sensing results. We also showed how to handle static defaults in the sense that these are applied every time after an action has been performed. Because of the way only-knowing is modelled, defaults behave as in autoepistemic logic. In previous work we showed that by modifying the semantics of only-knowing in the static case, other forms of default reasoning like Reiter's default logic can be captured (Lakemeyer and Levesque 2006). We believe that these results will carry over to our dynamic setting as well.

References

P. Blackburn, J. Kamps, and M. Marx, Situation calculus as hybrid logic: First steps. In P. Brazdil and A. Jorge (Eds.) *Progress in Artificial Intelligence,* Lecture Notes in Artificial Intelligence 2258, Springer Verlag, 253-260, 2001.

Claßen, J.; Eyerich, P.; Lakemeyer, G.; and Nebel, B. 2007. Towards an integration of Golog and planning. In Veloso, M. M., ed., *Proc. of IJCAI-07,* 1846–1851.

Fikes, R., and Nilsson, N. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2:189–208.

G. De Giacomo and M. Lenzerini, PDL-based framework for reasoning about actions, *Proc. of AI*IA'95,* LNAI 992, 103–114, 1995.

R. Demolombe, Belief change: from Situation Calculus to Modal Logic. *IJCAI Workshop on Nonmonotonic Reasoning, Action, and Change (NRAC'03),* Acapulco, Mexico, 2003.

R. Demolombe, A. Herzig, and I.J. Varzinczak, Regression in modal logic, *J. of Applied Non-Classical Logics*, 13(2):165-185, 2003.

H. Enderton, *A Mathematical Introduction to Logic,* Academic Press, New York, 1972.

Michael Gelfond and Vladimir Lifschitz. Representing action and change by logic programs. *Journal of Logic Programming,* 17:301–321, 1993.

D. Harel, Dynamic Logic, in D. Gabbay and F. Guenther (Eds.), *Handbook of Philosophical Logic*, Vol. 2, D. Reidel Publishing Company, 497–604, 1984.

A. Herzig, J. Lang, D. Longin, and T. Polacsek, A logic for planning under partial observability. In *Proc. AAAI-2000*, AAAI Press.

Hindriks, K. V.; De Boer, F. S.; Van der Hoek, W.; and Meyer, J.-J. C. 1999. Agent programming in 3APL. *Autonomous Agents and Multi-Agent Systems* 2(4):357–401.

G. Hughes, and M. Cresswell, *An Introduction to Modal Logic*, Methuen and Co., London, 1968.

A. Kakas, L. Michael, and R. Miller, Fred meets Tweety. In *Proc. ECAI, IOS Press,* 747–748, 2008.

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

G. Lakemeyer and H. J. Levesque, Situations, si! Situation Terms, no!. In *Ninth Conf. on Principles of Knowledge Representation and Reasoning,* AAAI Press, 2004.

G. Lakemeyer and H. J. Levesque, A useful fragment of the situation calculus. IJCAI-05, AAAI Press, 490–496, 2005.

G. Lakemeyer and H. J. Levesque, Towards an axiom system for default logic. In *Proc. of AAAI-06, AAAI Press,* 2006.

G. Lakemeyer and H. J. Levesque, A Semantical Account of Progression in the Presence of Defaults. in *Proc. IJCAI-09,* 2009.

G. Lakemeyer and H. J. Levesque, A Semantical Account of Progression in the Presence of Defaults (extended version). in *Conceptual Modeling: Foundations and Applications,* A. Borgida, V. Chaudhri, P. Giorgini, E. Yu (Eds.), Springer LNCS, 2009.

H. J. Levesque, All I Know: A Study in Autoepistemic Logic. *Artificial Intelligence*, **42**, 263–309, 1990.

H. J. Levesque and G. Lakemeyer, *The Logic of Knowledge Bases*, MIT Press, 2001.

H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl., Golog: A logic programming language for dynamic domains. *Journal of Logic Programming,* **31**, 59–84, 1997.

F. Lin and R. Reiter, How to progress a database, *Artificial Intelligence*, Elsevier, 92, 131-167, 1997.

Y. Liu and H. J.Levesque, 2005. Tractable reasoning with incomplete first-order knowledge in dynamic systems with context-dependent actions. In *Proc. of IJCAI-05*.

J. McCarthy and P. J. Hayes, Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer, D. Mitchie and M. Swann (Eds.) *Machine Intelligence 4,* Edinburgh University Press, 463–502, 1969.

R. C. Moore, Semantical considerations on nonmonotonic logic. *Artificial Intelligence* **25**, 75–94, 1985.

R. Reiter, The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation*, 359–380. Academic Press, 1991.

R. Reiter, *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems*. MIT Press, 2001.

M. Shanahan. *Solving the Frame Problem.* MIT Press, 1997.

R. B. Scherl and H. J. Levesque, Knowledge, action, and the frame problem. *Artificial Intelligence* 144(1-2), 1-39, 2003.

M. Thielscher. From situation calculus to fluent calculus: State update axioms as a solution to the inferential frame problem. *Artificial Intelligence,* 111(1–2):277–299, 1999.

S. Vassos and H. J. Levesque, 2007. Progression of situation calculus action theories with incomplete information. In *Proc. IJCAI-07*.

Parthood Simpliciter vs. Temporary Parthood

Claudio Masolo
Laboratory for Applied Ontology, ISTC-CNR
email: masolo@loa-cnr.it

Abstract

Starting from the formal characterization of four-dimensionalism (perdurantism) provided by Theodore Sider, I study the interconnections between the theories of *parthood simpliciter* (classical mereologies) and the theories of *temporary parthood* (parthood at a time). On the basis of this formal analysis, I propose a definition of temporary parthood in terms of parthood simpliciter that does not commit to the existence of temporal parts. In this way, I hope this definition can be accepted by endurantists.

Introduction

According to Sally Haslanger (Haslanger 2003, pp. 316–317), most of the puzzles about change through time rely on general conditions that, when integrally accepted, generate a contradiction. She individuates five general conditions:

1. Objects persist through change.
2. The properties involved in a change are incompatible.
3. Nothing can have incompatible properties.
4. The object before the change is one and the same object after the change.
5. The object undergoing the change is itself the proper subject of the properties involved in the change.

Let us consider, for example, a rose r that persists through the change from 'red' (R) to 'brown' (B), two incompatible properties, i.e., $\neg\exists x(R(x) \wedge B(x))$. Accepting the previous conditions, $R(r) \wedge B(r)$ holds, leading to a contradiction, that, to be solved, requires the rejection of (at least) one of the conditions (1)–(5).

In this paper I will focus on two positions on persistence through time, *perdurantism* and *endurantism*[1], that avoid the previous contradiction by rejecting, respectively, condition (5) and (2).

Perdurantism assumes that all the objects persist by *perduring*, i.e., similarly to the extension through space, objects are extended in time by having different (temporal) parts at different times. At each time, only a part of a persisting object is present, i.e. at one time persisting objects are only *partially present*. The subjects of temporal properties are temporal parts. In the previous example, 'r is P at t' must be read as 'r-at-t is P' where 'r-at-t' is the temporal part of r at t. Because r-at-t and r-at-t' are two different temporal parts of r (if $t \neq t'$), the contradiction disappears.

Endurantism assumes that some objects undergoing the change *endure*[2], i.e. they are *wholly present* at any time at which they exist, they maintain their identity through change and they are the subjects of properties, but these properties need to be *temporally qualified*. Red and brown are incompatible only if stated at the same time (about the same object), the fact that r is red-at-t and brown-at-t' does not lead to any contradiction. Different readings of 'P-at-t' are accepted by endurantists (e.g. modal or relational readings are considered, see (Varzi 2003)) that however refuse the applicability of the perdurantist view to all kinds of objects.

While the notion of *being partially present* has been quite precisely stated (Sider 1997; 2001), the notion of *being wholly present* is still quite obscure, even though some attempts to characterize it exist (Crisp and Smith 2005). This complicates the formal comparison between perdurantism and endurantism that often reduces to different positions on *parthood*: endurantists claim that, for enduring objects, a temporally qualified parthood (called here *temporary parthood*) is required, while perdurantists often refer to an atemporal parthood (called here *parthood simpliciter* or simply parthood) that is enough (together with a predicate of existence in time) to define temporal parts (see next section for the details).

To overcome this 'deadlock', Theodore Sider introduced a formal characterization of perdurantism based on temporary parthood (Sider 1997; 2001). On one side, perdurantists are able to accept his formulation simply analyzing 'x is part of y at t' as 'the temporal part of x at t is part of the temporal part of y at t'. On the other side, he hopes that the formalization of perdurantism in terms of temporary parthood can be 'intelligible' by endurantists.

In this paper, I don't provide a characterization of endurantism, I will just show that endurantists do not neces-

[1] I prefer to use the terms 'endurantim' and 'perdurantism' instead of *three-* and *four-dimensionalism*, because I will concentrate on persistence through time ignoring the spatial dimension. All the results are valid in any n-dimensional space-time (with $n \geq 2$).

[2] Usually endurantists also accept perduring objects, e.g. *processes* or *events*, as opposed to endurants, e.g. persons or cars.

sarily need to consider *temporary parthood* as primitive. I will prove that the axioms for temporary parthood can be 'recovered' in a theory based on parthood simpliciter *without* assuming the existence of temporal parts. This requires a new definition of temporary parthood (see (d9)) in terms of parthood simpliciter (and existence in time) that does not rely on temporal parts. Endurantists could accept this formulation analyzing 'x is part (simpliciter) of y' just as *constant parthood*, i.e. 'at every time at which x exists, x is part of y'. I think that this analysis prevents an a priori refutation of having parthood simpliciter as primitive and it offers an alternative to the usual tensed interpretation that reduces 'part-of' to 'part-of, now'. In addition to that, I formally analyze the interconnections between theories of parthood and theories of temporary parthood and how these interconnections depend on existential conditions (about the entities in the domain), a particularly important aspect to uncover the ontological commitment of perdurantism and endurantism.

One may wonder if a deeper understanding of perdurantism and endurantism is relevant for representing commonsense knowledge. I do not have a definite answer, but only few considerations. First, perdurantism is not incompatible with commonsense. Commonsense and natural language are deeply related and perdurantism offers an alternative ontological foundation to the semantics of natural language that can handle a number of well-known semantic phenomena (Muller 2007). Second, Patrick Hayes, in his seminal work (Hayes 1985), already encountered the problem of understanding temporal parts: the ontological status of the couples $\langle objects, time \rangle$ he uses has not been clarified. Third, my analysis is quite general and can be helpful in formalizing different domains. For example some qualitative theories of space-time and movement are based on four-dimensional entities (Muller 1998). Fourth, perdurantism has recently been adopted in some applications not only to overcome some technical difficulties (as in the case of the representation of n-ary relations in *description logics* (Welty and Fikes 2006)) but also advocating its adequateness, conceptual simplicity and practical advantages for representing dynamic environments (Stell and West 2004; West 2004).

Formal characterization of perdurantism

Following Sider, temporal existence is represented by the primitive $\mathsf{EX}xt$ whose informal reading is "at time t, x exists". I'm concerned here with persistence through time, therefore I focus only on objects that *are* in time, objects that *exist* at some times:

a1 $\exists t(\mathsf{EX}xt)$

EX has to be intended just as a *representational surrogate* that does not necessarily commit neither on the existence/nature of times nor on the fact that existence is an extrinsic relation between objects and times. Times could be constructed from events like in (Kamp 1979) or just be the reification of the worlds of a (modal) temporal logic. Existence in time can be reduced to 'being simultaneous with' others entities (Simons 1991) or, assuming a Newtonian view in which time is an independent container, to a *location* relation. Times can be punctual or extended and different structures (discrete vs. continuous, linear vs. branching, etc.) can be imposed on them. For the purpose of this paper, time can just be considered as a set of indexes, i.e. a set of atomic entities that are related only by *identity*.

The notion of *parthood simpliciter* is represented by the predicate $\mathsf{P}xy$ that can be read as "x is part of y".

On the basis of parthood simpliter and existence in time, the (perdurantist) notion of *temporal part* (also called *temporal slice*) can be defined. x is a temporal part of y at t, formally $\mathsf{TP}xyt$, if x is a maximal part of y that exists *only* at t. Formally (using the relation $\mathsf{O}xy$ defined in (d1) that stands for "x overlaps y"):

d1 $\mathsf{O}xy \triangleq \exists z(\mathsf{P}zx \wedge \mathsf{P}zy)$

d2 $\mathsf{TP}xyt \triangleq \mathsf{EX}xt \wedge \mathsf{EX}yt \wedge \neg\exists t'(\mathsf{EX}xt' \wedge t' \neq t) \wedge \mathsf{P}xy \wedge \forall z(\mathsf{P}zy \wedge \mathsf{EX}zt \rightarrow \mathsf{O}zx)$

Following the schema adopted by perdurantists for all the temporary properties and relations, *temporary parthood* ($\mathsf{tP}xyt$ stands for "x is part of y at t") can be defined as:

d3 $\mathsf{tP}xyt \triangleq \exists zw(\mathsf{TP}zxt \wedge \mathsf{TP}wyt \wedge \mathsf{P}zw)$

Because endurantists accept objects that do not necessarily have temporal parts at every time at which they exist, they refuse (d3) as a *general* definition of temporary parthood. For *enduring* objects temporary parthood has to be taken as primitive, or an alternative to (d3) that does not rely on temporal parts needs to be provided.

Sider's formulation

In (Sider 1997; 2001), Sider proposes a formulation of perdurantism based on the primitive of *temporary parthood* instead of parthood simpliciter. He hopes that this move can lead to a theory 'intelligible' both to perdurantists and endurantists, allowing for a formal comparison of the two positions.

The axioms and definitions considered by Sider are reported below (see (Sider 2001, pp. 58–59)) where $\mathsf{tO}xyt$ stands for "x overlaps y at t", and $\mathsf{tTP}xyt$ stands for "x is a temporal part of y at t"[3]:

d4 $\mathsf{tO}xyt \triangleq \exists z(\mathsf{tP}zxt \wedge \mathsf{tP}zyt)$

d5 $\mathsf{tTP}xyt \triangleq \neg\exists t'(\mathsf{EX}xt' \wedge t' \neq t) \wedge \mathsf{tP}xyt \wedge \forall z(\mathsf{tP}zyt \rightarrow \mathsf{tO}zxt)$

a2 $\mathsf{tP}xyt \rightarrow \mathsf{EX}xt \wedge \mathsf{EX}yt$

a3 $\mathsf{EX}xt \rightarrow \mathsf{tP}xxt$

a4 $\mathsf{tP}xyt \wedge \mathsf{tP}yzt \rightarrow \mathsf{tP}xzt$

a5 $\mathsf{EX}xt \wedge \mathsf{EX}yt \wedge \neg\mathsf{tP}xyt \rightarrow \exists z(\mathsf{tP}zxt \wedge \neg\mathsf{tO}zyt)$

Sider characterizes perdurantism (four-dimensionalism in his vocabulary) as:

[3] I use different symbols to represent the temporal part relation defined in terms of temporary parthood (d5) from the one defined in terms of parthood simpliciter (d2).

> “[N]ecessarily, each spatiotemporal object has a temporal part at every moment at which it exists.” (Sider 2001, p. 59)[4]

This claim seems a restriction of the one given in (Sider 1997, p. 206) where Sider refers to objects in *time* instead of in *space-time*. In the original work of Lesniewski (Lesniewski 1991) mereology is not intended as a theory necessarily related to space or space-time but as a pure formal theory (that applies to all kinds of entities) aimed at avoiding some (ontological) assumptions of set-theory, namely, the existence of the *empty set* and the distinction between *urelements* and *sets*. In this sense mereology does not commit to existence in space or time. Even though a theory of persistence must consider entities in time, I do not see any reason to exclude entities that (according to some researchers) do not have a clear spatial location, e.g. mental attitudes, concepts, mathematical theories, societies. I thus prefer the following characterization:

> “Each object that exists in time has a temporal part at every time at which it exists.”, i.e. formally:

pd $\mathsf{EX}xt \rightarrow \exists y(\mathsf{tTP}yxt)$

$\mathcal{T}_{\mathsf{tP}} = \{$(a1)–(a5), (pd)$\}$ denotes Sider's theory where tO and tTP are respectively defined by (d4) and (d5).

Theorem (t1) shows that at a given time, the temporal parts are not necessarily unique. A counterexample is provided by a model with two *different* elements a and b, both existing only at t, such that $\langle a, a, t\rangle$, $\langle b, b, t\rangle$, $\langle a, b, t\rangle$, $\langle b, a, t\rangle \in \mathsf{tP}^{\mathcal{I}}$. In this case, it is easy to verify that $\langle a, a, t\rangle$, $\langle b, a, t\rangle \in \mathsf{tTP}^{\mathcal{I}}$.

(t2) shows that different entities can coincide (they are part one of the other) during their whole life. The previous model is a counterexample because both a and b exist only at t and $\langle a, b, t\rangle, \langle b, a, t\rangle \in \mathsf{tP}^{\mathcal{I}}$, but $a \neq b$ by hypothesis.

t1 $\mathcal{T}_{\mathsf{tP}} \nvdash \mathsf{tTP}xyt \wedge \mathsf{tTP}zyt \rightarrow x = z$

t2 $\mathcal{T}_{\mathsf{tP}} \nvdash \forall t(\mathsf{EX}xt \rightarrow \mathsf{tP}xyt) \wedge \forall t(\mathsf{EX}yt \rightarrow \mathsf{tP}yxt) \rightarrow x = y$

Formulation based on parthood simpliciter

Sider shows that P and EX allow to define the notions of *temporal part* and *temporary parthood* (respectively by (d2) and (d3)) and to characterize perdurantism by an axiom similar to (pd). However, Sider does not clarify what axioms on P and EX are necessary to have a theory *equivalent* to $\mathcal{T}_{\mathsf{tP}}$. I intend equivalence in the following way: (i) all the axioms in $\mathcal{T}_{\mathsf{tP}}$ can be proved in this new theory by *assuming* the ‘same’ EX and the definition (d3) for tP; (ii) the new theory does not add new properties on tP and EX.

According to (Simons 1987) and (Casati and Varzi 1999), parthood is minimally characterized as a *partial order*, i.e., a reflexive, antisymmetric, and transitive binary relation (axioms (a6), (a7), and (a8)). The inclusion of the *extensionality* (axiom (a9)) guarantees the identity of objects that have the same parts (t3) or that overlap the same objects (t4). The theory $\mathcal{M}_{\mathcal{E}} = \{$(a6)–(a9)$\}$ is called *extensional mereology*.

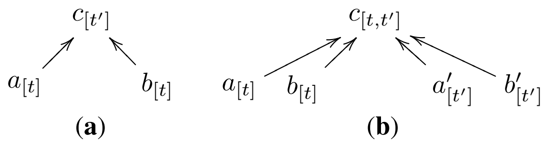

Figure 1: (pdn) is independent from (a10).

a6 $\mathsf{P}xx$

a7 $\mathsf{P}xy \wedge \mathsf{P}yx \rightarrow x = y$

a8 $\mathsf{P}xy \wedge \mathsf{P}yz \rightarrow \mathsf{P}xz$

a9 $\neg\mathsf{P}xy \rightarrow \exists z(\mathsf{P}zx \wedge \neg\mathsf{O}zy)$

t3 $\mathcal{M}_{\mathcal{E}} \vdash \forall z(\mathsf{P}zx \leftrightarrow \mathsf{P}zy) \rightarrow x = y$

t4 $\mathcal{M}_{\mathcal{E}} \vdash \forall z(\mathsf{O}zx \leftrightarrow \mathsf{O}zy) \rightarrow x = y$

It is possible to characterize perdurantism by introducing an axiom analogous to (pd):

pdn $\mathsf{EX}xt \rightarrow \exists y(\mathsf{TP}yxt)$

Because, as already observed, P can in general apply to all kinds of objects, standard mereologies do not analyze how P and EX are related. (pdn) is a weak link between P and EX that does not rule out models like the one in figure 1.a where some of the parts of c (namely, a and b) have temporal extensions disjoint from the one of c.[5]

(a10) rules out these models by ensuring that the temporal extension of the part is included in the one of the whole. First of all note that (a10) and (pdn) are independent: the model in figure 1.a satisfies (pdn) but not (a10) and vice versa for the model in figure 1.b. Secondly, and more importantly, by defining parthood simpliciter as *constant parthood* (d6), (t5) shows that, in $\mathcal{T}_{\mathsf{tP}}$, (a10) holds. Therefore, assuming (d6), the lack of (a10) prevents any equivalence between $\mathcal{T}_{\mathsf{tP}}$ and the theory based on parthood simpliciter we are building.

a10 $\mathsf{P}xy \wedge \mathsf{EX}xt \rightarrow \mathsf{EX}yt$

d6 $\mathsf{P}xy \triangleq \forall t(\mathsf{EX}xt \rightarrow \mathsf{tP}xyt)$

t5 $\mathcal{T}_{\mathsf{tP}} \vdash_{(\mathrm{d6})} \{$(a10)$\}$

Let $\mathcal{T}_{\mathsf{P}} = \{$(a1),(a6)–(a10),(pdn)$\}$, where O, TP, and tP are defined by (d1)–(d3).

(t6) shows that $\mathcal{T}_{\mathsf{P}}$ is at least as strong as $\mathcal{T}_{\mathsf{tP}}$, i.e., all the axioms in $\mathcal{T}_{\mathsf{tP}}$ can be proved in $\mathcal{T}_{\mathsf{P}}$ by assuming the same EX and the definition (d3) for tP.

(t7) and (t8) show that $\mathcal{T}_{\mathsf{P}}$ is strictly stronger than $\mathcal{T}_{\mathsf{tP}}$, because in $\mathcal{T}_{\mathsf{tP}}$ temporal parts (at a specific time) are not unique and different entities can coincide (see (t1) and (t2)).

[4] Sider does not explicitly introduce a modal operator in his formulation. However note that in a classical first order logic all the formula can be considered as ‘necessary’.

[5] The graphical notation adopted follows four conventions: (i) the times at which an entity exists are subscribed between square brackets; (ii) an arc from a to b without labels stands for parthood, i.e., $\langle a, b\rangle \in \mathsf{P}^{\mathcal{I}}$; (iii) an arc from a to b with label t stands for temporary parthood at t, i.e., $\langle a, b, t\rangle \in \mathsf{tP}^{\mathcal{I}}$; (iv) all the arcs due to reflexivity and transitivity closure of parthood are omitted. For example the graph in figure 1.a depicts the following model: $D = \{a, b, c, t, t'\}$, $\mathsf{EX}^{\mathcal{I}} = \{\langle a, t\rangle, \langle b, t\rangle, \langle c, t'\rangle\}$, and $\mathsf{P}^{\mathcal{I}} = \{\langle a, a\rangle, \langle b, b\rangle, \langle c, c\rangle, \langle a, c\rangle, \langle b, c\rangle\}$.

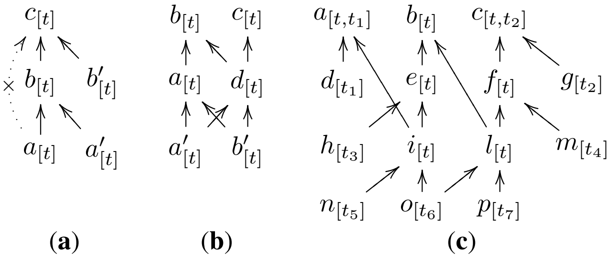

Figure 2: Counterexamples to the transitivity of the temporary parthood.

t6 $\mathcal{T}_{\mathsf{P}} \vdash_{(\mathrm{d3})} \mathcal{T}_{\mathsf{tP}}$

t7 $\mathcal{T}_{\mathsf{P}} \vdash_{(\mathrm{d3})} \mathsf{tTP}yxt \wedge \mathsf{tTP}yzt \rightarrow y = z$

t8 $\mathcal{T}_{\mathsf{P}} \vdash_{(\mathrm{d3})} \forall t(\mathsf{EX}xt \rightarrow \mathsf{tP}xyt) \wedge \forall t(\mathsf{EX}yt \rightarrow \mathsf{tP}yxt) \rightarrow x = y$

To find a theory based on P equivalent to $\mathcal{T}_{\mathsf{tP}}$ it is then necessary to weaken $\mathcal{T}_{\mathsf{P}}$.

t9 $\mathcal{T}_{\mathsf{P}} \smallsetminus \{(\mathrm{a8})\} \nvdash_{(\mathrm{d3})} (\mathrm{a4})$

t10 $\mathcal{T}_{\mathsf{P}} \smallsetminus \{(\mathrm{a9})\} \nvdash_{(\mathrm{d3})} (\mathrm{a4})$

t11 $\mathcal{T}_{\mathsf{P}} \smallsetminus \{(\mathrm{a10})\} \nvdash_{(\mathrm{d3})} (\mathrm{a4})$

t12 $\mathcal{T}_{\mathsf{P}} \smallsetminus \{(\mathrm{a7})\} \vdash_{(\mathrm{d3})} \mathcal{T}_{\mathsf{tP}}$

t13 $\mathcal{T}_{\mathsf{P}} \smallsetminus \{(\mathrm{a7})\} \nvdash \mathsf{TP}yxt \wedge \mathsf{TP}zxt \rightarrow y = z$

t14 $\mathcal{T}_{\mathsf{P}} \smallsetminus \{(\mathrm{a7})\} \nvdash_{(\mathrm{d3})} \mathsf{tTP}yxt \wedge \mathsf{tTP}zxt \rightarrow y = z$

t15 $\mathcal{T}_{\mathsf{P}} \smallsetminus \{(\mathrm{a7})\} \nvdash_{(\mathrm{d3})} \forall t(\mathsf{EX}xt \rightarrow \mathsf{tP}xyt) \wedge \forall t(\mathsf{EX}yt \rightarrow \mathsf{tP}yxt) \rightarrow x = y$

t16 $\mathcal{T}_{\mathsf{tP}} \vdash_{(\mathrm{d6})} \mathcal{T}_{\mathsf{P}} \smallsetminus \{(\mathrm{a7})\}$

(t9), (t10), and (t11) show that weakening $\mathcal{T}_{\mathsf{P}}$ by respectively dropping the transitivity, the extensionality or the 'temporal monotonicity' of P lead to a too weak theory in which the transitivity of the temporary parthood (defined via (d3)) does not hold: figures 2.a, 2.b, and 2.c respectively illustrate a model of $\mathcal{T}_{\mathsf{P}} \smallsetminus \{(\mathrm{a8})\}$, $\mathcal{T}_{\mathsf{P}} \smallsetminus \{(\mathrm{a9})\}$, and $\mathcal{T}_{\mathsf{P}} \smallsetminus \{(\mathrm{a10})\}$ in which $\langle a, b, t\rangle, \langle b, c, t\rangle \in \mathsf{tP}^{\mathcal{I}}$ but $\langle a, c, t\rangle \notin \mathsf{tP}^{\mathcal{I}}$ (in figure 2.a, the curved arrow on the left makes explicit that in this case the transitivity closure is not valid, i.e. we have $\langle a, c\rangle \notin \mathsf{P}^{\mathcal{I}}$).

(t12)–(t15) show that, dropping the antisymmetry of parthood, the embedding is maintained but the uniqueness of TP and tTP does not holds and it is possible to have different coincident objects. As a counterexample, let us consider: $\mathsf{EX}^{\mathcal{I}} = \{\langle a, t\rangle, \langle b, t\rangle\}$, $\mathsf{P}^{\mathcal{I}} = \{\langle a, a\rangle, \langle b, b\rangle, \langle a, b\rangle, \langle b, a\rangle\}$.

(t16) shows that $\mathcal{T}_{\mathsf{tP}}$ can be embedded in $\mathcal{T}_{\mathsf{P}} \smallsetminus \{(\mathrm{a7})\}$ via (d6). In addition, it is possible to prove that expanding the definition of P in terms of the vocabulary of $\mathcal{T}_{\mathsf{tP}}$, and, successively expanding the formula obtained using the definition of tP given in $\mathcal{T}_{\mathsf{P}}$, we re-obtain P; similarly starting from the expansion of the definition of tP in terms of the vocabulary of $\mathcal{T}_{\mathsf{P}}$. Therefore $\mathcal{T}_{\mathsf{P}} \smallsetminus \{(\mathrm{a7})\}$ and $\mathcal{T}_{\mathsf{tP}}$ are *equivalent*.

It is also possible to strengthen $\mathcal{T}_{\mathsf{tP}}$ via (a11) (an axiom that directly corresponds to the antisymmetry of P) to prove the equivalence between $\mathcal{T}_{\mathsf{P}}$ and $\mathcal{T}_{\mathsf{tP}} \cup \{(\mathrm{a11})\}$.

a11 $\forall t(\mathsf{EX}xt \rightarrow \mathsf{tP}xyt) \wedge \forall t(\mathsf{EX}yt \rightarrow \mathsf{tP}yxt) \rightarrow x = y$

The two equivalences and the theorems (t1) and (t2) show that the main difference between $\mathcal{T}_{\mathsf{tP}}$ and $\mathcal{T}_{\mathsf{P}}$ concerns the uniqueness of the temporal parts and the acceptance of the *coincidence* of different objects (different objects that are one part the other during their whole life).

These topics have been deeply discussed in the literature on (material) *constitution* (see (Rea 1997) for a good review). According to (a11), if, for example, the clay that constitutes a statue and the statue itself are different, they cannot coincide during their whole life (even though the distinction is based on a difference in modal behavior).[6] From my point of view, this represents a genuine difference between perdurantism and endurantism. While perdurantists, identifying coincidence with identity, tend to reduce differences among objects to mereological ones (in particular spatio-temporal ones), endurantists tend to accept coincidence between different objects motivating this distinction by, not necessarily mereological, different temporary property. While perdurantists have a multiplicative approach towards parts, endurantists have a multiplicative approach towards coincident objects.

Avoiding temporal parts

In this section, I introduce an alternative definition of temporary parthood in terms of parthood and I show which existential conditions are necessary to embed the theory based on parthood in the one based on temporary parthood.

Let us start observing that, as showed by (t17) and (t18), the equivalence between $\mathcal{T}_{\mathsf{P}} \smallsetminus \{(\mathrm{a7})\}$ and $\mathcal{T}_{\mathsf{tP}}$ and the one between $\mathcal{T}_{\mathsf{P}}$ and $\mathcal{T}_{\mathsf{tP}} \cup \{(\mathrm{a11})\}$ both rely on the existence of temporal parts.

t17 $\mathcal{T}_{\mathsf{P}} \smallsetminus \{(\mathrm{pdn})\} \nvdash_{(\mathrm{d3})} (\mathrm{a3})$

t18 $\mathcal{T}_{\mathsf{tP}} \cup \{(\mathrm{a11})\} \smallsetminus \{(\mathrm{pd})\} \nvdash_{(\mathrm{d6})} (\mathrm{a9})$

The situation in figure 1.b is a model of $\mathcal{T}_{\mathsf{P}} \smallsetminus \{(\mathrm{pdn})\}$ in which $\langle c, t\rangle \in \mathsf{EX}^{\mathcal{I}}$ and $\langle c, c, t\rangle \notin \mathsf{tP}^{\mathcal{I}}$ (because c has no temporal parts).

$\mathsf{EX}^{\mathcal{I}} = \{\langle a, t\rangle, \langle b, t\rangle, \langle b, t'\rangle\}$, $\mathsf{tP}^{\mathcal{I}} = \{\langle a, a, t\rangle, \langle b, b, t\rangle, \langle a, b, t\rangle, \langle b, a, t\rangle\}$ is a model of $\mathcal{T}_{\mathsf{tP}} \cup \{(\mathrm{a11})\} \smallsetminus \{(\mathrm{pd})\}$ in which $\langle b, a\rangle \notin \mathsf{P}^{\mathcal{I}}$ but, because a is part of itself and it is also part of b, both a and b overlap a, i.e.$\langle a, a\rangle, \langle a, b\rangle \in \mathsf{O}^{\mathcal{I}}$. This situation fails to satisfy (a9) because the only part (simpliciter) of b different from b (a *proper* part of b) is a that does not exists at t'. Therefore (a5) does not introduce any new object because it applies neither at t (a and b coincide at t) nor at t' (only b exists at t'). (pd) allows to prove (a9) by introducing the temporal part of b at t'.

According to (t17), by refusing (pdn), endurantists cannot accept (d3) as a general definition of tP in terms of P. In the following, I propose an alternative definition that commits to existential conditions weaker than (pdn). More specifically, I consider an *extensional closure mereology* (Casati and Varzi 1999) extended just with (a10), i.e. the theory

$\mathcal{T}^{c}_{\mathsf{P}} = \{(\mathrm{a1}), (\mathrm{a6})\text{–}(\mathrm{a10}), (\mathrm{a12}), (\mathrm{a13})\}$,

[6] Interpreting parthood as spatio-temporal inclusion, (a11) excludes the possibility of having spatio-temporally co-located entities.

where SUM ($\mathsf{SUM}sxy$ stands for "s is a sum of x and y") and DIF ($\mathsf{DIF}dxy$ stands for "d is a difference between x and y") are defined by (d7)–(d8). Note that to avoid 'empty objects', according to (a13), the difference between x and y exists only in case x is not part of y.

d7 $\mathsf{SUM}sxy \triangleq \forall z(\mathsf{O}zs \leftrightarrow \mathsf{O}zx \vee \mathsf{O}zy)$

d8 $\mathsf{DIF}dxy \triangleq \forall z(\mathsf{P}zd \leftrightarrow \mathsf{P}zx \wedge \neg\mathsf{O}zy)$

a12 $\exists s(\mathsf{SUM}sxy)$

a13 $\neg\mathsf{P}xy \rightarrow \exists d(\mathsf{DIF}dxy)$

(d9) is my alternative to (d3). Informally, (d9) may be explained in the following way: let us suppose that both x and y exist at t, then x is part of y at t if and only if (i) x is part of y at every time at which it exists (and therefore, in particular, at t); or (ii) if x is part of y only during a part of its life (the life of x), then this part of life includes t. The condition (ii) can be restated: if x is not part of y at t (and x exists at t), then the difference between x and y exists at t because some parts of x that exist at t are not part of y.

(t19) allows for interpreting parthood as constant part.

d9 $\mathsf{tP}xyt \triangleq \mathsf{EX}xt \wedge \mathsf{EX}yt \wedge (\mathsf{P}xy \vee \exists d(\mathsf{DIF}dxy \wedge \neg\mathsf{EX}dt))$[7]

t19 $\mathcal{T}^c_{\mathsf{P}} \vdash_{(\mathrm{d9})} \mathsf{P}xy \leftrightarrow \forall t(\mathsf{EX}xt \rightarrow \mathsf{tP}xyt)$

t20 $\mathcal{T}^c_{\mathsf{P}} \nvdash_{(\mathrm{d9})}$ (a4)

(t20) shows that $\mathcal{T}^c_{\mathsf{P}}$ is too weak. Figure 3 depicts[8] a situation in which $\langle A, B\rangle, \langle B, C\rangle \in \mathsf{tP}^{\mathcal{I}}$ but $\langle A, C\rangle \notin \mathsf{tP}^{\mathcal{I}}$. To understand why, let us note that $\langle a, A, B\rangle$, $\langle b, B, C\rangle$, $\langle A, A, C\rangle \in \mathsf{DIF}^{\mathcal{I}}$, but only A (that is the only difference between A and C) exists at t and this fact prevents the possibility of having $\langle A, C, t\rangle \in \mathsf{tP}^{\mathcal{I}}$. Notice that A exists at t even though all its proper parts (a and b) exist only at t'.

Therefore to embed $\mathcal{T}^c_{\mathsf{P}}$ in $\mathcal{T}_{\mathsf{tP}} \cup \{(\mathrm{a11})\} \smallsetminus \{(\mathrm{pd})\}$ we need to strengthen $\mathcal{T}^c_{\mathsf{P}}$. (t22), (t23) and (t24) show that (a14) (from which (t21) follows directly) does the job without committing to the existence of temporal parts. A situation with only one object that exists at two different times is a simple counterexample to both (pdn) and (pd), but it is possible also to build complex counterexamples following the situations in figure 4.

a14 $\mathsf{DIF}dxy \wedge \mathsf{EX}xt \wedge \neg\mathsf{EX}yt \rightarrow \mathsf{EX}dt$

t21 $\mathsf{SUM}sxy \wedge \mathsf{EX}st \rightarrow (\mathsf{EX}xt \vee \mathsf{EX}yt)$

t22 $\mathcal{T}^c_{\mathsf{P}} \cup \{(\mathrm{a14})\} \vdash_{(\mathrm{d9})} \mathcal{T}_{\mathsf{tP}} \cup \{(\mathrm{a11})\} \smallsetminus \{(\mathrm{pd})\}$

t23 $\mathcal{T}^c_{\mathsf{P}} \cup \{(\mathrm{a14})\} \nvdash$ (pdn)

t24 $\mathcal{T}^c_{\mathsf{P}} \cup \{(\mathrm{a14})\} \nvdash_{(\mathrm{d9})}$ (pd)

Without committing to temporal parts, $\mathcal{T}^c_{\mathsf{P}} \cup \{(\mathrm{a14})\}$ and the definition (d9) offers endurantists the possibility to choose parthood simpliciter as primitive, informally reading this relation as 'constant parthood' which, in my understanding, does not violate any endurantist principle.

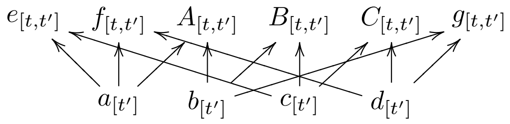

Figure 3: Counterexample to the transitivity of tP.

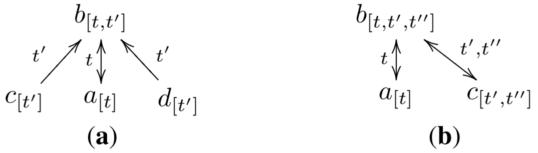

Figure 4: Counterexamples to (pd).

It is clear that the equivalence between $\mathcal{T}^c_{\mathsf{P}} \cup \{(\mathrm{a14})\}$ and $\mathcal{T}_{\mathsf{tP}} \cup \{(\mathrm{a11})\} \smallsetminus \{(\mathrm{pd})\}$ cannot be proved. Strongly, (t18) shows that $\mathcal{T}_{\mathsf{tP}} \cup \{(\mathrm{a11})\} \smallsetminus \{(\mathrm{pd})\}$ is too weak to prove the extensionality of P via (d6).

This last problem can be solved by substituting $\{(\mathrm{pd}), (\mathrm{a5})\}$ with (a15): $\mathcal{T}^n_{\mathsf{tP}} = \{(\mathrm{a1})\text{–}(\mathrm{a4}), (\mathrm{a15})\}$. (t25), (t26), and (t27) show $\mathcal{T}^n_{\mathsf{tP}} \cup \{(\mathrm{a11})\}$ does not commit to temporal parts but it is strong enough to 'recover' the extensionality of P. In addition, (t28) shows that $\mathcal{T}^n_{\mathsf{tP}} \cup \{(\mathrm{a11})\}$ is not too strong with respect to $\mathcal{T}^c_{\mathsf{P}} \cup \{(\mathrm{a14})\}$.

a15 $\mathsf{EX}xt \wedge \neg\mathsf{tP}xyt \rightarrow \exists z(\mathsf{tP}zxt \wedge \forall t'(\neg\mathsf{tO}zyt'))$

t25 $\mathcal{T}^n_{\mathsf{tP}} \vdash$ (a5)

t26 $\mathcal{T}^n_{\mathsf{tP}} \cup \{(\mathrm{a11})\} \nvdash$ (pd)

t27 $\mathcal{T}^n_{\mathsf{tP}} \cup \{(\mathrm{a11})\} \vdash_{(\mathrm{d6})} \mathcal{T}_{\mathsf{P}} \smallsetminus \{(\mathrm{pdn})\}$

t28 $\mathcal{T}^c_{\mathsf{P}} \cup \{(\mathrm{a14})\} \vdash_{(\mathrm{d9})} \mathcal{T}^n_{\mathsf{tP}} \cup \{(\mathrm{a11})\}$

Figure 4 depicts two counterexamples to (pd) in $\mathcal{T}^n_{\mathsf{tP}} \cup \{(\mathrm{a11})\}$. The example in figure 4.a does not satisfy the 'maximality' imposed to temporal parts, while in the example in figure 4.b the fact that objects need to have a temporal part at any time at which they exist does not hold. However, the existential commitment imposed by (a15) is inevitably stronger than the one imposed by (a5).

However, the embedding of $\mathcal{T}^n_{\mathsf{tP}} \cup \{(\mathrm{a11})\}$ in $\mathcal{T}^c_{\mathsf{P}} \cup \{(\mathrm{a14})\}$ does not hold, i.e. $\mathcal{T}^c_{\mathsf{P}} \cup \{(\mathrm{a14})\}$ is strictly stronger than $\mathcal{T}^n_{\mathsf{tP}} \cup \{(\mathrm{a11})\}$. In particular, the existential commitment provided by (a15) is too weak to guarantee the existence of the difference (a13). Informally, (a13) requires that (i) the difference between x and y is a part of x and (ii) that all the objects that are part of x and do not overlap y are part of the difference. These conditions are not imposed by (a15). The example in figure 5 does not satisfy the above condition (ii) but it satisfies (a15) (and all the other axioms of $\mathcal{T}^n_{\mathsf{tP}} \cup \{(\mathrm{a11})\}$): at t, a is not part of b, but c satisfies the constraint in (a15). At t, a is not part of c, but b satisfies the constraint in (a15). At t', a is neither part of b nor c, but d satisfies the constraint in (a15) in both cases. Because a is present at t but, at this time, a is not part of b, then by (d6), a is not part simpliciter of b, then the hypothesis of (a13) is

[7] In an extensional closure mereology, if x is not part of y then the difference exists and it is *unique*, therefore (d9) is equivalent to $\mathsf{tP}xyt \triangleq \mathsf{EX}xt \wedge \mathsf{EX}yt \wedge (\neg\mathsf{P}xy \rightarrow \forall d(\mathsf{DIF}dxy \rightarrow \neg\mathsf{EX}dt))$.

[8] For the sake of conciseness, in the figure are reported only the sums of couples of atomic objects. The graph needs to be completed with the sums of three and four atomic objects that however are not relevant for the proof of (t20).

$$\begin{tikzcd}
 & a_{[t,t']} \arrow[dr, leftrightarrow, "t'"] & \\
b_{[t]} \arrow[ur, "t"] & c_{[t]} \arrow[u, "t"] & d_{[t']}
\end{tikzcd}$$

Figure 5: Counterexample to (a13).

satisfied and the existence of the difference between a and b must be proved. The only candidates for the difference are c and d. c cannot be the difference because d is part of a at t', d does not overlap b at any time, but d is not part of c at any time. d cannot be the difference because c is part of a at t, c does not overlap b at any time, but c is not part of d at any time.

In addition to that, in $\mathcal{T}^n_{\mathsf{tP}} \cup \{(\text{a11})\}$ nothing guarantees the existence of sums, for example models with two objects, one existing only at t, the other one existing only at t' are not ruled out.

Conclusions and further work

In this work I studied some interconnections between theories based on parthood simpliter and theories based on temporary parthood. I showed that, to build a theory based on parthood simpliciter equivalent to the theory of Sider, the antisymmetry of parthood cannot be included. I analyzed how this result can explain some divergences between endurantism and perdurantism. In addition to that, theorem (t19) and theorems (t22)–(t24), together with (d9), make explicit the possibility to have a characterization of temporary parthood in terms of parthood simpliciter that, without committing to the existence of temporal parts, may be accepted by endurantists (at least from my point of view).

Some formal results are still lacking. In particular I do not know how the theory $\mathcal{T}^n_{\mathsf{tP}} \cup \{(\text{a11})\}$ can be extended in order to prove the equivalence with $\mathcal{T}^c_{\mathsf{P}} \cup \{(\text{a14})\}$. A straightforward possibility consists in adding to $\mathcal{T}^n_{\mathsf{tP}} \cup \{(\text{a11})\}$ the analogue of axioms (a12) and (a13), but the proof of equivalence with $\mathcal{T}^c_{\mathsf{P}} \cup \{(\text{a14})\}$ is not trivial. Another open problem concerns the independence of the existence of differences from (a15) plus the existence of sums: in presence of (a15), is the existence of sums enough to guarantee the existence of differences?

Finally, I think that, at least in the case of applications, one of the main motivations to follow a perdurantist approach concerns the possibility to reduce the predication of a property an object has at t, to the predication on the temporal part of the object at t. By avoiding temporal parts, my analysis does not provide any alternative to this reduction. I think that a possible alternative compatible with the endurantist view is offered by *trope theory* (see (Daly 1997) for a good survey) that conceives change as trope substitution. But this theory, that in any case commits to a new kind of objects called tropes, requires other basic primitives notions as *inherence* and *resemblance* that cannot be grasped only in terms of parthood.

Acknowledgments. I'm very grateful to Laure Vieu and to the anonymous reviewers for their really helpful comments.

References

Casati, R., and Varzi, A. 1999. *Parts and Places. The Structure of Spatial Representation*. Cambridge, MA: MIT Press.

Crisp, T. M., and Smith, D. P. 2005. Wholly present defined. *Philosophy and Phenomenological Research* 71:318–344.

Daly, C. 1997. Tropes. In Mellor, D., and Oliver, A., eds., *Properties*. Oxford: Oxford University Press. 140–159.

Haslanger, S. 2003. Persistence through time. In Loux, M. J., and Zimmerman, D. W., eds., *The Oxford Handbook of Metaphysics*. Oxford: Oxford University Press. 315–354.

Hayes, P. 1985. Naive physics 1: Ontology for liquids. In Hobbs, J., and Moore, R., eds., *Formal Theories of the Commonsense World*. Norwood: Ablex. 71–108.

Kamp, H. 1979. Events, istants and temporal reference. In Baüerle, R.; Egli, U.; and Stechow, A. v., eds., *Semantics from Different Points of View*. Berlin: Springer. 376–417.

Lesniewski, S. 1991. *Collected Works*. Dordrecht: Kluwer.

Muller, P. 1998. A qualitative theory of motion based on spatio-temporal primitives. In Cohn, A. G.; Schubert, L.; and Shapiro, S. C., eds., *International Conference on Principles of Knowledge Representation and Reasoning (KR 98)*, 131–141. Trento, Italy: Morgan Kaufmann.

Muller, P. 2007. The temporal essence of spatial objects. In Aurnague, M.; Hickmann, M.; and Vieu, L., eds., *The Categorization pf Spatial Entities in Language and Cognition*. John Benjamins Publishing Co. 285–306.

Rea, M., ed. 1997. *Material Constitution. A Reader*. Rowman and Littlefield: Lanham, MD.

Sider, T. 1997. Four-dimensionalism. *The Philosophical Review* 106(197-231).

Sider, T. 2001. *Four-Dimensionalism. An Ontology of Persistence and Time*. Oxford: Clarendon Press.

Simons, P. 1987. *Parts: a Study in Ontology*. Oxford: Clarendon Press.

Simons, P. 1991. On being spread out in time: temporal parts and the problem of change. In Spohn, W. e. a., ed., *Existence and Explanation*. Kuwer Achademic Publishers. 131–147.

Stell, J. G., and West, M. 2004. A 4-dimensionalist mereotopology. In Varzi, A., and Vieu, L., eds., *Formal Ontology in Information Systems (FOIS04)*, 261–272. Turin: IOS Press.

Varzi, A. C. 2003. Riferimento, predicazione, e cambiamento. In Bianchi, C., and Bottani, A., eds., *Significato e ontologia*. Milano: Franco Angeli. 221–249.

Welty, C., and Fikes, R. 2006. A reusable ontology for fluents in owl. In Varzi, A. C., and Vieu, L., eds., *Proceedings of FOIS-06. Baltimore, USA.*, 226–236. IOS Press.

West, M. 2004. Some industrial experiences in the development and use of ontologies. In *EKAW04, Workshop on Core Ontologies*.

Evaluation of EPILOG: a Reasoner for Episodic Logic

Fabrizio Morbini and **Lenhart Schubert**
University of Rochester

Abstract

It can be quite hard to objectively evaluate a reasoner geared towards commonsense problems and natural language applications if it uses a nonstandard logical language for which there exist no publicly available datasets. We describe here the evaluation of our recent improvements of the EPILOG system, a reasoner for Episodic Logic, a superset of first-order logic geared towards natural language applications. We used both a sample of interesting commonsense questions obtained from the ResearchCyc knowledge base and the standard TPTP library to provide an evaluation that tests the unique features of Episodic Logic and also puts the performance of EPILOG into perspective with respect to the state of the art in first-order logic theorem provers. The results show the extent of recent improvements to EPILOG, and that very expressive commonsense reasoners need not be grossly inefficient.

Introduction

We present here the evaluation of the progress made in the development of the EPILOG system ((Schubert et al. 1993) and (Schaeffer et al. 1993)), motivated by the recent effort towards building a self-aware agent (Morbini and Schubert 2008). EPILOG is an inference engine for Episodic Logic (EL) ((Schubert and Hwang 2000) and (Hwang and Schubert 1993)) that has been under development since 1990 ((Schubert et al. 1993) and (Schaeffer et al. 1993)).

The EPILOG system and EL are designed with natural language (NL) understanding in mind. The natural way to test its capabilities (both on the reasoning front and on the representation front) is by using a publicly available set of commonsense problems.

Among several collections that are available, we opted for the set of problems contained in the ResearchCyc knowledge base. They comprise more than 1600 problems that provide both the English formulation of a question and its translation into CycL[1]. In addition to the abundance of interesting and challenging questions, another advantage of using this dataset is that it allows the comparison between our and Cyc's interpretation of each question.

The last point highlights the problem of comparison for systems that use for their evaluation a dataset based on English. Because a question expressed in English can be formalized in many ways and at various levels of detail, it is very difficult to use the results obtained to compare different systems. This lack of a dataset expressed in logic to facilitate comparisons is not easily solved given the lack of agreement on a single logical language well-suited for NL; and even if such a language existed each English sentence can still be interpreted in many ways and at different levels of detail.

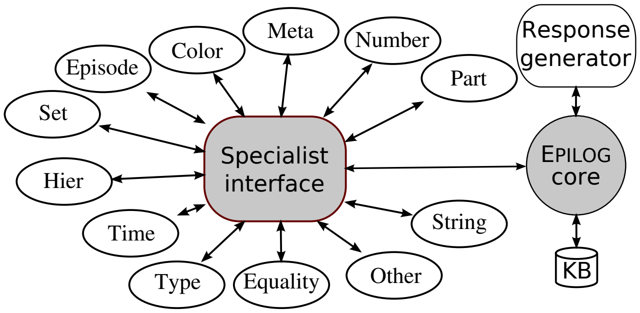

Figure 1: The high level structure of EPILOG1.

Therefore, to give a more complete picture of the performance of the EPILOG system and to facilitate comparisons with other systems, we decided to evaluate it as well against the widely used TPTP dataset for FOL theorem provers. This puts the basic performance of the reasoner in perspective with respect to the state of the art in FOL theorem provers. The evaluation on Cyc's commonsense test cases instead tests the features that distinguish EPILOG from a traditional FOL theorem prover.

In the paper if we need to distinguish between the legacy EPILOG system and the new version we will refer to the former as EPILOG1 and to the latter as EPILOG2.

This paper is organized as follows: first we briefly describe the high-level structure of the EPILOG system, and then highlight the major improvements made to EPILOG in the EPILOG2 system. Then we describe in detail the evaluation of the system and state our conclusions.

EPILOG

In this section we briefly describe EPILOG and EL. Figure 1 represents the building blocks of the EPILOG1 system and

[1] http://www.cyc.com/cycdoc/ref/cycl-syntax.html

how they are connected together. EPILOG1's core contains the inference routines, the parser, the normalizer and the storage and access schemas to retrieve and add knowledge from/to the knowledge base. A set of specialists, connected to the core inference engine through an interface module, help the general inference routines to carry out special inferences quickly (e.g., type inference to conclude whether [*Car1 Artifact*] is true given that *Car1* is a coupe and coupes are a type of car, cars are vehicles, and vehicles are artifacts). The specialist interface consists of a series of flags associated with some key predicates/functions that automatically activate predefined functions in a particular specialist.

EPILOG is a reasoner for EL. EL is a highly expressive natural logic with unique features, including modifiers, reifiers, substitutional and generalized quantifiers and episodic operators, making EL particularly suited for (NL) applications. Briefly, the major differences with respect to FOL are the following.

To represent events and their relations, three *episodic operators* are introduced: *, ** and @. These operators take a well-formed formula (wff) and a term (an event) as arguments. For example, the EL formula [[*D1 lose-control-of V1*] ** *e1*] expresses that *e1* is the event characterized by *D1* losing control of *V1*. (Note that predicates are preceded by their "subject" argument in wffs.) *Substitutional quantification* over predicative expressions, wffs, and other syntactic entities is required to express meaning postulates and introspective knowledge. It is also important for interfacing the general inference engine with specialists (as described later). *EL modifiers* correspond to the modifiers used in NL, e.g., "very", "almost" or "by sheer luck", and reification operators are used to represent generics and attitudes. *Quantifiers* allow for the use of a restrictor. For example, in the sentence "Most dogs are friendly", "dogs" is the restrictor of the quantifier "most". For the quantifiers ∀ and ∃ the restrictor can be incorporated into the remainder of the quantified sentence, but for many generalized quantifiers this is not possible.

EPILOG2

In this section we mention the major changes made to EPILOG1. The interface to knowledge bases (KB) has been redesigned to facilitate **1)** temporary modifications to a KB (introduced for example by the assumption-making used during inference) and **2)** the development and testing of new access schemas (i.e. mechanisms to retrieve knowledge from a KB). The result is a KB system based on inheritance of KBs (similar to what Cyc uses for inheritance of microtheories) in which each KB is associated with a particular access schema that can be easily changed.

The parser was changed from an if-then based mechanism to a system based on a standard chart parser. This allows for easy debugging and modifications to the ever-evolving EL grammar.

The interface to specialists is now based on explicit metaknowledge stored like any other knowledge. This knowledge specifies under what conditions a particular specialist functionality can be called. For example the formula ($\forall_{wff}$ w ['w without-free-vars] [[(apply 'apply-fn-knownbyme? 'w) = 'yes] ⇒ [(that w) knownbyme]]) describes when the introspective specialist can be called to answer whether EPILOG knows a particular formula *w*. The interface is based on this *Apply* function, which is known to the inference engine as having a special meaning.

An automatic system to extract type information has been added to EPILOG. Currently this system is used **1)** to build type hierachies, **2)** to keep track of the return type of functions based on the type of the arguments and **3)** to build a hierarchy for the arguments of transitive predicates (also transitive predicates are automatically detected by looking for formulas like (∀ x (∀ y (∀ z (([x P y] and [y P z]) ⇒ [x P z])))), expressing transitivity).

The question-answering (QA) framework has been totally redesigned to allow for QA inside QA (used in introspection and called *recursive QA*). In addition subgoals are now selected using a hierachical agenda that sorts the subgoals based on **1)** the size of the formula associated with subgoal g relative to the size of the biggest formula among the siblings of g; **2)** the % of times a descendant of g or g itself was selected for inference but no improvement was obtained[2]; **3)** the % of g that is solved (this is greater than 0 only for a subgoal that at some point can be split, e.g., a conjunction); **4)** the % difference between the size of g's formula and the size of the smallest formula among the descendants of g whose solution would imply a solution of g; for conjunction of subgoals, their average size is considered.

Evaluation

To evaluate the progress of our effort to build a self-aware agent based on EPILOG2, we used two methods: **1)** testing on a selected small set of examples from the commonsense test cases contained in Research Cyc; **2)** the scalability test included in the TPTP library of problems for theorem provers; this scalability test was constructed from the OpenCyc knowledge base. With the first type of evaluation we are testing the adequacy of EL for directly expressing English questions and background knowledge, and the reasoning capabilities of EPILOG2. With the second type of evaluation we are testing how EPILOG2 fares in relation to the state of the art of FOL theorem provers.

First we will describe the set of questions used to test EPILOG2's commonsense reasoning capabilities. Most of the questions have been manually encoded in EL because the general-purpose English to EL translator is not yet robust enough to handle these questions. However care has been taken not to simplify the EL form of those questions to make the job of the reasoner easier; instead we made an effort to produce EL versions that would likely be produced by an automatic, compositional English-to-EL translator. This is why some questions may appear more complex than one might expect, based on traditional "intuited" formalizations of English sentences.

In the formulas used in the following examples, we use Epi2Me as the internal constant that refers to the system itself.

[2] An improvement is measured either by a decrease in size of the resulting subgoal, or solution of the subgoal.

Question 1 is *“How old are you?”*, which in EL becomes:

(wh$_{term}$ x ($\exists_{term}$ y [’x rounds-down ’y]
(∃ z [’y expresses z (K (plur year))]
(∃ e [e at-about Now]
[[z age-of Epi2Me] ** e]))))

K is a reification operator that maps a predicate (here, (plur year), a predicate true of any collection of years) to a *kind* (here, the kind whose realizations are collections of years).

We have assumed that the representation of the question would be expanded pragmatically to include conventional restrictions on the form of the answer expected, i.e., an answer in rounded-down years rather than, say, seconds. These pragmatic constraints depend on the question itself; for example they would be different for a question like “How old is this bagel/star/rock/etc.?”. In the future we would like to automatically include such constraints by means of “cooperative conversation axioms”. We might have an axiom saying something like: *If X informs Y about a quantitative attribute F (such as weight, age, temperature, etc.) of some entity Z, then X is conversationally obligated to express F(Z) in units that are conventional for entities of the type(s) instantiated by Z.* In addition we would need various axioms about the conventional units for expressing weight, age, etc., of various types of entities. These axioms would then be used to refine the raw logical form of a question to include the pragmatic constraints. However, here we just focused on solving the question, manually adding the necessary pragmatic constraints.

Some of the key knowledge used to answer this question is the following:

This axiom defines the age of an entity during a particular event, when the entity’s birth date is known:

(∀ y (∀ x [x (be (birth-date-of y))]
(∀ e [[(time-elapsed-between (date-of e) x) age-of y] @ e])))

Axiom defining the relation between the ** and @ operators:

($\forall_{wff}$ w (∀ e ([w @ e] ⇔
(∃ e1 [e1 same-time e] [w ** e1]))))

Axiom that describes which specialist function to call to express the function *time-elapsed-between* in a particular type of unit:

(∀ x (x is-date) (∀ y (y is-date)
($\forall_{pred}$ type (’type el-time-pred)
(∀ r (’r = (Apply ’diff-in-dates? ’x ’y ’type))
(’r expresses (time-elapsed-between x y)
(K (plur type)))))))

The most interesting part of this example is the use of a set of axioms based on the *Apply* function to make the reasoning system “aware” of a set of procedures useful in computing mathematical operations and in doing type conversions. In this way EPILOG2 is able to return the answer to the question expressed as an integer that is the floor of the amount of time in years that has elapsed between the date of birth of EPILOG and now (the moment of speech). In EL the unifier found for the variable x of the initial question is: (amt 18 (K (plur year))).

Question 2 is *“What’s your name?”*, which expressed in EL is:

(∃ e [e at-about now0]
[(wh z ([z name] and [Epi2Me have z])
(∃ y [y thing] [y (BE (L x (x = z)))])) ** e])

Some of the key knowledge used to answer this question is the following:

The event now0 is during the event e2:

[now0 during e2]

The event e2 is characterized by EPILOG having the name ’epilog-name’:

[[Epi2Me have ’epilog-name] ** e2]

If one event is characterized by something possessing something else, then that will also be true for any event during the first event:

(∀ x (∀ y (∀ z [[x have y] ** z]
(∀ zz [zz during z] [[x have y] @ zz]))))

Of interest here is the last axiom because it ascribes “inward persistence” (homogeneity) to predicate *have*, a property it shares with other *atelic* predicates. The two other formulas are hand-additions to the current knowledge base, but they should be automatically inserted, the first by the English to EL generator, the second by a self-awareness demon that is in charge of maintaining basic information about the agent, for instance, its name, its state (e.g. sleeping, awake, etc.) and its “state of health” (e.g., cpu consumption, free memory, garbage collection status, etc.).

To correctly answer this question the reasoner also uses lexical knowledge that states which predicates are atemporal and therefore can be moved out of the scope of the ** operator. This knowledge is expressed in EL and it is used by the normalizer. An example is (’thing EL-type-pred), stating that ’thing’ is a type predicate and therefore atemporal.

Question 3 shows how EPILOG could answer questions about its own knowledge. The question is *“What do you know about the appearance of pigs?”*, which in EL we expressed as:

(wh x [x appearance-fact-about (K (plur pig))])

Some of the relevant knowledge involved in this example is:

Pigs are thick-bodied:

[(K (plur pig)) thick-bodied]

The predicate ‘thick-bodied’ is an appearance predicate:

[’thick-bodied appearance-pred]

Every wff that uses an appearance predicate is a fact about the appearance of its subject:

($\forall_{pred}$ p [’p appearance-pred]
(∀ x [x p] [(that [x p]) appearance-fact-about x]))

One could construct much more complex formulas pertaining to the appearance of something, e.g., that the appearance of a person’s hair – say, color and style – constitutes appearance information about the person.

The remaining questions are taken from the ResearchCyc 1.0 collection of commonsense test cases. About 81% of these test cases have been axiomatized to become solvable by Cyc; among those presented here, the last two have a solution in Cyc. An important difference between our and Cyc's approach to these problems is in the style of formalization: Cyc's representations are in a simplified form that **1)** is geared towards the CycL style (e.g., using many concatenated names for complex expressions instead of compositionally combining the parts), which is far from NL-based representations; and **2)** omits important details (e.g. temporal relations) and pragmatic constraints.

Question 4 is *"Can gasoline be used to put out a fire?"*. In Cyc this is the test case named `#$CST-CanYouUseGasToPutOutAFire`, and the question is expressed as: `((TypeCapableFn behaviorCapable) GasolineFuel ExtinguishingAFire instrument-Generic)`. `(TypeCapableFn behaviorCapable)` returns a predicate that describes the capacity for a certain behavior of a certain type of thing in a certain role position. In effect the question becomes, "Is gasoline-fuel behaviorally-capable of being a generic-instrument in fire-extinguishing?"

We also interpret the question generically, but we adhere more closely to a possible English phrasing, asking whether there could be an instance where a person uses gasoline to put out a fire:

(∃ e [e during (extended-present-rel-to Now)]
 (∃ x [x person]
  (∃ y [y ((nn gasoline) fuel)]
   (∃ z [z fire]
    [[x (able-to ((in-order-to (put-out z)) (use y)))]
     @ e]))))

Some of the knowledge relevant to this question is:

If some person is able to use some stuff to put-out a fire then s/he must be at the same location as the fire, must have at hand that stuff and that stuff must be flame-suppressant:

(∀ e [e during (extended-present-rel-to Now)]
 (∀ x [x person] (∀ y [y stuff] (∀ z [z fire]
  ([[x (able-to ((in-order-to (put-out z)) (use y)))] @ e]
   ⇒ ([[x has-at-hand y] @ e] ∧ [[x loc-at z] @ e]
    [y flame-suppressant]))))))

Gasoline is flammable stuff:

(∀ x [x ((nn gasoline) fuel)] ([x flammable] ∧ [x stuff]))

Flammable things are not flame-suppressant:

(∀ x [x flammable] (not [x flame-suppressant]))

The question is answered negatively by using the knowledge that to be able to put-put a fire one must use a flame-suppressant material, and gasoline is not a flame-suppressant material.

Question 5 is Cyc's question named `#$CST-DoesCycHaveABiologicalFather`, which in English is *"Do you (Cyc) have a biological father?"*. In Cyc the question is represented as `(thereExists ?F (biologicalFather Cyc ?F))`.

We expressed the question in EL as follows:

(∃ e [e at-about Now]
 (∃ y [[Epi2Me (have-as ((attr biological) father)) y]
   ** e]))

In this question, *have-as* is a so-called "subject-adding operator" that takes a unary predicate as argument and returns a binary predicate. In this case ((attr biological) father) is the monadic predicate true for all individuals that are biological fathers. (have-as ((attr biological) father)) is the binary predicate that is true for all pairs of individuals in which the object of the predicate is the father of its subject.

The relevant knowledge for this example is:

EPILOG is an artifact:

[Epi2Me artifact]

No artifact is a natural object:

(∀ x [x artifact] (not [x natural-obj]))

A creature is a natural object:

(∀ x [x creature] [x natural-obj])

All creatures have a biological father:

(∀ x ([x creature] ⇔
  (∃ y (∃ e
   [[x (have-as ((attr biological) father)) y] ** e]))))

The question is answered negatively by using the knowledge that EPILOG is an artificial thing and therefore not a natural object. Further it is known that only creatures can have a biological father and that creatures are a subtype of natural objects.

Question 6 corresponds to Cyc's question named `#$CST-AnimalsDontHaveFruitAsAnatomicalParts-HypothesizedQueryTest` In Cyc the question is expressed as `(implies (isa ?ANIMAL Animal) (not (relationInstanceExists anatomicalParts ?ANIMAL Fruit)))`.

In EL we express the question (more naturally, we claim) as:

(∀ e [e during (extended-present-rel-to Now)]
  (No x [x animal]
   [[x (have-as anatomical-part) (K fruit)] ** e]))

The function *extended-present-rel-to* applied to an event e returns the event that started long ago and continues long pass the end of the event e. The extent of the event returned should be context-dependent. However, for this question this is irrelevant given that the knowledge used is presumed true for any event. The relevant knowledge for this example is:

Plant stuff is not animal stuff:

(∀ x [x plant-stuff] (not [x animal-stuff]))

Fruits are made of plant stuff:

[(K fruit) made-of (K plant-stuff)]

Animals are made of animal stuff:

[(K animal) made-of (K animal-stuff)]

If an individual x is made of (kind of stuff) p and if (kind of stuff) q is a subtype of p then x is made of q:

(∀ x ($\forall_{pred}$ p [x made-of (k p)]
 ($\forall_{pred}$ q (∀ y [y p] [y q])
   [x made-of (k q)])))

If an individual x is made of (kind of stuff) p and if (kind of stuff) q is disjoint from p then x is not made of q:

(∀ x ($\forall_{pred}$ p [x made-of (k p)]
($\forall_{pred}$ q (∀ y [y p] (not [y q]))
(not [x made-of (k q)]))))

If a type p is made of (kind of stuff) q then all individuals of type p are made of q:

($\forall_{pred}$ p ($\forall_{pred}$ q ([(k p) made-of (k q)] ⇔
(∀ y [y p] [y made-of (k q)]))))

Every part is made of the material of the whole:

(∀ w (∀ e (∀ p
([[w (have-as anatomical-part) p] ** e] ⇒
(∀ wm [w made-of wm] [p made-of wm])))))

We decided to answer the question by saying that all parts are made of the same substance of which the whole is made. However the case of artificial parts/organs is not captured by this knowledge. One could improve on it by saying that organic parts must be made of biologically compatible materials, while any artificial parts must be made of durable inert materials that are compatible with the organic parts they are in contact with.

Question 7 corresponds to Cyc's question named `#$CST--DoAgentsBelieveWhatTheyKnow`. The English version of the question reads *"If you know that something is the case, do you believe that it is the case?"*. In Cyc the question is represented as: `(implies (knows ?AGT ?PROP) (beliefs ?AGT ?PROP))`. In EL we provide the following representation as a direct reflection of English surface form[3]:

(∀ e0 [e0 at-about Now]
(∀ x [x thing]
([[Epi2Me know (that (∃ e1 [e1 at-about e0]
[[x (be the-case)] ** e1]))
] ** e0] ⇒
(∃ e2 ([e2 at-about Now] and [e0 same-time e2])
[[Epi2Me (believe
(that (∃ e3 [e3 at-about e2]
[[x (be the-case)] ** e3])))
] ** e2]))))

The key knowledge to answer this question is the following axiom:

If an event is characterized by some agent knowing something then it is also characterized by the agent believing it:

(∀ e (∀ x (all p ([[x know p] ** e] ⇒ [[x believe p] ** e]))))

Question 8 (our last example) corresponds to Cyc's commonsense test case named `#$CST-CanYouAttack-SomeoneWithAGolfClub`. In English the question is *"Can you attack someone with a golf club?"*. Cyc expresses it in the same way as question 4: `((TypeCapableFn behaviorCapable) GolfClub Physically-AttackingAnAgent deviceUsedAsWeapon)`.

In EL we represent the question as:[4]

(∃ x [x golf-club] (∃ y [y person] (∃ z [z person]
(∃ e [[y ((adv-a (with-instr x)) (attack z))] ** e]))))

The knowledge relevant to this question is:

If an object can be swung by hand, and is solid, and weighs at least two pounds, it can be used as a striking weapon:

(∀ x [x phys-obj]
[[(∃ e [[x (pasv ((adv-a (by (k hand))) swing))] ** e]) ∧
[x solid]
(∃ w [[x weighs w] ∧ [w ¿= (k ((num 2) pound))]])]
⇒ (∃ e [[x (pasv (use-as ((nn striking) weapon)))] ** e])])

A golf club can be swung by hand, is solid, and weighs at least two pounds:

(∀ x [x golf-club]
[(some e [[x (pasv ((adv-a (by (k hand))) swing))] ** e]) ∧
[x solid] [x phys-obj] (∃ w [[x weighs w] ∧
[w ≥ (k ((num 2) pound))]])])

For any striking weapon, one person can attack another with the weapon, by striking him or her with it:

(∀ x [x ((nn striking) weapon)] (∃ y [y person] (∃ z [z person]
(∃ e [[y ((adv-a (by-means (Ka ((adv-a (with-instr x))
(strike z)))))
((adv-a (with-instr x)) (attack z)))] ** e]))))

There is a golf-club:

(∃ x [x golf-club])

(”by-means” modification is monotone) If an agent does some action by means of another action, then he does the first action:

($\forall_{pred}$ p (∀ x (∀ y
(∀ e [[x ((adv-a (by-means y)) p)] ** e] [[x p] ** e]))))

This question is answered positively by using the knowledge that golf-clubs are heavy and solid and can be swung by a person and that objects with those properties can be used to attack another person.

FOL scalability tests: the second part of the evaluation put into perspective the performance of the reasoner with respect to standard FOL theorem provers on the classic TPTP[5] dataset. In particular we used the CSR[6] problems derived from the conversion into FOL of the OpenCyc ontology (Ramachandran, Reagan, and Goolsbey 2005).

We used the subset of CSR problems that was designed to test the scalability of a theorem prover. In particular the problems used were those designated as `CSR025` through `CSR074` in segments 1 to 5. Even though the access schema of EPILOG2 is a simple exhaustive one and therefore not scalable, the results will provide a good bottom-line comparison with future improvements of EPILOG.

Table 1 summarizes the results. The systems compared are EPILOG1[7], EPILOG2, and Vampire 9, which is represen-

[3] apart from the events and event relations introduced by the temporal deindexing that follows logical form computation (Schubert and Hwang 2000).

[4] EPILOG also answers the case in which "you" is interpreted literally to mean EPILOG itself. In this case, the question is answered negatively using introspection, a closure axiom that asserts that EPILOG's knowledge with respect to major abilities is complete, the fact that physical actions are major ability and that attacking somebody requires the ability to perform physical actions.

[5] See http://www.cs.miami.edu/~tptp/

[6] See http://www.opencyc.org/doc/tptp_challenge_problem_set in particular the section *The Scaling Challenge Problem Set*.

[7] In particular it is the version of June 22^{nd}2005.

Segment	Size (min/avg/max)	EPILOG1 FI	EPILOG1 no FI	EPILOG2	Avg depth	Vampire 9
1	(22/59/163)	46	46	100	5.9	100
2	(-/1101/-)	46	44	92	5.6	100
3	(-/7294/-)	0	0	54	4.5	82
4	(-/42981/-)	0	0	48	4.3	32
5	(-/534435/-)	0	0	12	1.3	0

Table 1: Summary of the tests carried out between EPILOG1, EPILOG2 and the Vampire theorem prover, version 9. The first column contains the segment number (1-5) of the segments comprising the scalability subset of the CSR dataset (with 50 problems in each segment). Column 2 lists min, max and average number of formulas contained in the problems in that specific segment. (If all problems contain the same number of formulas only the average is shown). Columns 3, 4, and 5 show the percentage of problems for which a solution was found, respectively by EPILOG1 with forward inference enabled, EPILOG1 without forward inference and EPILOG2 (which by default has no forward inference enabled). Column 6 shows the average depth of the answer found by EPILOG2. Column 7 shows the percentage of problems solved by Vampire. All system have been limited to a timeout of 120 seconds.

tative of state-of-the-art FOL theorem provers[8]. All systems were run under the same conditions and were subjected to a 2 minute limit per problem.

Conclusion and Further Work

In this paper we described how we evaluated the work on the development of the latest version of the EPILOG system in a way that we think tests the particular features that characterize EPILOG and that also may allow for comparison with other commonsense reasoners independently of which logical language they use.[9]

The evaluation was divided into 2 parts. In the first we selected 8 examples, five of which were from ResearchCyc. These examples were selected to test the features of EL and of EPILOG such as introspective question answering, quotation and subtitutional quantification, interfacing to specialists, etc. The second part was based on a subset of the TPTP dataset used to test the scalability of a theorem prover. This part, in addition to providing a baseline for assessing future enhancements of EPILOG, demonstrates significant performance gains achieved here over EPILOG1, and will facilitate further comparisons with other theorem provers. Moreover, the results show that a reasoner for a highly expressive logic doesn't have to be impractically inefficient compared to a less expressive one[10]. It should be kept in mind that in addition to not lagging far behind state-of-the-art performance in FOL theorem provers in their domain of competence, EPILOG is capable of additional modes of reasoning and metareasoning as shown by the first evaluation.

In future we plan to close the remaining gap between EPILOG and FOL theorem provers, implement a more efficient access schema for knowledge retrieval, implement probabilistic reasoning, provide for uniform handling of generalized quantifiers, and extend the new approach to specialist deployment to all specialists.

Acknowledgements

This work was supported by NSF grant IIS-0535105 and by a 2007-2008 gift from Bosch Research and Technology Center (Palo Alto); the content has benefited significantly from the very useful comments of the anonymous referees.

References

Hwang, C., and Schubert, L. 1993. Episodic logic: A situational logic for natural language processing. In P. Aczel, D. Israel, Y. K., and Peters, S., eds., *Situation Theory and its Applications*, volume 3. Stanford, CA: Center for the Study of Language and Information. 303–338.

Morbini, F., and Schubert, L. K. 2008. Metareasoning as an integral part of commonsense and autocognitive reasoning. In *Metareasoning 08*, 155–162.

Ramachandran, D.; Reagan, P.; and Goolsbey, K. 2005. First-Orderized ResearchCyc: Expressivity and Efficiency in a Common-Sense Ontology.

Schaeffer, S.; Hwang, C.; de Haan, J.; and Schubert, L. 1993. EPILOG, the computational system for episodic logic: User's guide. Technical report, Dept. of Computing Science, Univ. of Alberta.

Schubert, L., and Hwang, C. 2000. Episodic Logic meets Little Red Riding Hood: A comprehensive, natural representation for language understanding. In Iwanska, L., and Shapiro, S., eds., *Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language*. Menlo Park, CA: MIT/AAAI Press. 111–174.

Schubert, L. K.; Schaeffer, S.; Hwang, C. H.; and de Haan, J. 1993. *EPILOG: The Computational System for Episodic Logic. USER GUIDE.*

[8] Download available at http://www.cs.miami.edu/~tptp/CASC/J4/Systems.tgz

[9] Allowing longer times had minimal effect on both systems.

[10] contrary to the alleged "expressivity/tractability tradeoff".

A BDI Agent Architecture for a POMDP Planner

Gavin Rens[1,2]
Alexander Ferrein[3]
Etienne van der Poel[1]
[1] School of Computing, Unisa, Pretoria, South Africa
[2] Knowledge Systems Group, Meraka Institute, CSIR, Pretoria, South Africa
[3] Robotics and Agents Research Laboratory, University of Cape Town, South Africa
grens@csir.co.za, alexander.ferrein@uct.ac.za, evdpoel@unisa.ac.za

Abstract

Traditionally, agent architectures based on the Belief-Desire-Intention (BDI) model make use of *pre-compiled* plans, or if they do *generate* plans, the plans do not involve stochastic actions nor probabilistic observations. Plans that *do* involve these kinds of actions and observations are generated by partially observable Markov decision process (POMDP) planners. In particular for POMDP planning, we make use of a POMDP planner which is implemented in the robot programming and plan language Golog. Golog is very suitable for integrating beliefs, as it is based on the situation calculus and we can draw upon previous research on this. However, a POMDP planner on its own cannot cope well with dynamically changing environments and complicated goals. This is exactly a strength of the BDI model; the model is for reasoning over goals dynamically. Therefore, in this paper, we propose an architecture that will lay the groundwork for architectures that combine the advantages of a POMDP planner written in the situation calculus, and the BDI model of agency. We show preliminary results which can be seen as a proof of concept for integrating a POMDP into a BDI architecture.

Introduction

Traditionally, plan-based agents that include generative planning (as opposed to utilizing pre-compiled plans) would generate a complete plan to reach a *specific fixed* goal, then execute the plan. If plan execution monitoring is available, the agent would replan from scratch when the plan becomes invalid. Due to the time requirements for generating complete plans, the plan may be invalid by the time it is executed. This is because the world may change substantially during plan generation.

Therefore, Belief-Desire-Intention (BDI) architectures take a different approach. BDI theory is based on the philosophy of practical reasoning (Bratman 1987). It offers flexibility in planning beyond traditional planning for agents, by reasoning over *different goals*. That is, an agent based on BDI theory can adapt to changing situations by focusing on the pursuit of the most appropriate goal at the time. Typically, an appropriate plan to achieve an adopted goal is then selected from a data base of plans. Although a plan that satisfies certain constraints (e.g., does not conflict with other adopted plans, is executable, etc.) will be adopted, it may not be the most appropriate plan in existence. A plan that is generated with the agent's current knowledge for guidance, may be more appropriate. BDI agents can also make rational decisions as to when to replan if a plan becomes invalid, reducing the amount of replanning, thus increasing the agent's reactivity. Note that the BDI model is, however, not the only approach to replanning (cf. (Likhachev et al. 2005)).

In general, BDI architectures do not make use of plan generation, they rather draw on plan libraries. While with BDI approaches, an agent can reason over several goals, the agent lacks some flexibility by not being able to generate suitable plans on demand. Therefore, in this paper, we aim at integrating a POMDP planner into a BDI architecture to combine its benefits with the ability to *generate* plans. Moreover, we want to supply models that are as realistic as possible. We therefore decided on employing partially observable Markov Decision Processes (POMDPs).

In this paper we describe our approach for combining BDI theory with a POMDP planner. Combining the two formalisms can be viewed from two perspectives. One, to enhance an existing planner for use in real-time dynamic domains by incorporating the planner into a BDI agent architecture so that the management of goal selection, planning and replanning is handled in a principled way. Two, to enhance the classical BDI agent architecture by incorporating a POMDP planner into the BDI architecture so that the agent can reason (plan) with knowledge about the uncertainty of the results of its actions, and about the uncertainty of the accuracy of its perceptions. We employ the POMDP planner described in our previous work (Rens, Ferrein, and Van der Poel 2008). This planner is implemented in Golog (Levesque et al. 1997), which in turn is based on the situation calculus (McCarthy 1963; Reiter 2001). An advantage of using a Golog implementation for the planner is that the integration of beliefs into the situation calculus has previously been done (e.g., (Bacchus, Halpern, and Levesque 1999)) and this work can be used for formulating POMDPs. Further, given a background action theory, an initial state and a goal state (or reward function in POMDPs), Golog programs essentially constrain and specify the search space (the space of available actions).

The resulting plan (or *policy* in POMDPs) is a Golog program which can be executed directly by the agent. To the best of our knowledge, till present, no BDI-based agent architecture has implemented its planning function so as to generate plans that take stochastic action and partial observation into account. Therefore, this work can be seen as a first proof of this concept.

The rest of the paper is organized as follows. In the next section we introduce the plan generator used in this study. Then, we briefly introduce the BDI theory, after which we explain our hybrid BDI/POMDP-planner architecture in detail. Before we conclude, we show some preliminary results from an implementation of our architecture, which gives a first proof of our approach.

The Planning Module

The POMDP Model

In partially observable Markov decision processes (POMDPs) actions have nondeterministic results, yet may be predicted with a probability of occurrence. And observations are uncertain: the world is not directly observable, therefore the agent *infers* how likely it is that the world is in some specific state. The agent thus believes to some degree—for each possible state—that it is in that state. Furthermore, a POMDP is a *decision* process and thus facilitates making decisions as to which actions to take, given its previous observations and actions. Formally, a POMDP is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O}, b_0 \rangle$ with: $\mathcal{S}$, a finite set of states of the world; $\mathcal{A}$, a finite set of actions; $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Pi(\mathcal{S})$ is the *state-transition function*, where Π is a probability distribution; $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, the *reward function*; Ω, a finite set of observations the agent can experience; $\mathcal{O} : \mathcal{S} \times \mathcal{A} \to \Pi(\Omega)$, the *observation function*; and b_0, the initial probability distribution over all world states in $\mathcal{S}$ (see e.g., (Kaelbling, Littman, and Cassandra 1998)). In the model, b is a *belief state*, i.e. a set of pairs (s, p) where each state $s \in \mathcal{S}$ is associated with a probability p. The *state estimation* function $SE(b, a, o)$ updates the agent's beliefs. Now the aim of the agent deploying a POMDP model is to determine a policy, that is, the actions or decisions that will maximize its rewards. Formally, a policy π is a function from a set of belief states B to the set of actions: $\pi : B \to A$. That is, actions are *conditioned* on beliefs. This means that the agent takes its next decision not only based on a stochastic action model, but also on a stochastic observation model. In this sense, a policy can be represented as a *policy tree*, with nodes being actions and branches being observations.

Planning over Degrees of Belief

In this section we describe our POMDP planner, an extension to the decision-theoretic language, DTGolog (Boutilier et al. 2000).

DTGolog is based on Reiter's variant of the situation calculus (McCarthy 1963; Reiter 2001), a second-order language for reasoning about actions and their effects. According to this calculus, changes in the world are due only to actions, so that a situation is completely described by the history of actions starting in some initial situation—$do(a, s)$ is the term denoting the situation resulting from doing action a in situation s. Properties of the world are described by *fluents*, which are situation-dependent predicates and functions. For each fluent the user defines a successor state axiom specifying precisely which value the fluent takes on after performing an action. These, together with precondition axioms for each action, axioms for the initial situation, and foundational and unique names axioms, form a so-called *basic action theory* (Reiter 2001).

Decision-theoretic planning in DTGolog works roughly as follows. Given an input program that leaves open several action alternatives for the agent, the DTGolog interpreter generates an optimal policy. Formally, the interpreter solves a Markov Decision Process (MDP, cf. e.g., (Puterman 1994)) using the forward search value iteration method—searching (to a specified horizon) for the actions that will maximize the total expected reward. Programs are interpreted as follows: All possible outcomes of the intended nondeterministic, stochastic action are expanded. For each choice point, the action resulting in the optimal value at the particular point in the MDP, is determined. These values are calculated relative to the world situation associated with the point in the MDP. The policy is calculated with an optimization theory consisting of a reward and a transition function (cf. also (Boutilier et al. 2000)). The transition function describing transition probabilities between states of the Markov chain is given by Reiter's variant of the basic action theory formalized in the underlying situation calculus (McCarthy 1963; Reiter 2001). Formally, the $BestDo$ macro defines the process described above: it evaluates an input program and recursively builds an optimal policy.

The POMDP planner we use here is $BestDoPO$ (Rens, Ferrein, and Van der Poel 2008); an extension of $BestDo$ ($BestDo$ Partially Observable), which calculates an optimal policy for the partially observable case (Rens, Ferrein, and Van der Poel 2008). The main difference is that $BestDoPO$ operates on a belief state rather than on a world state. $BestDoPO(p, b, h, \pi, v, pr)$ takes as arguments a Golog program p, a belief state b and a horizon h, which determines the solution depth sought by the interpreter. The policy π as well as its value v and the success probability pr are returned. After a certain action a is performed and the associated observation o is perceived, the next belief state is determined via a belief state transition function (similar in vein to the state estimation function of the previous subsection, and the successor-state axiom for likelihood weights as given in (Bacchus, Halpern, and Levesque 1999)):

$$\begin{aligned}
&b_{new} = BU(o, a, b) \doteq \\
&\quad b_{temp} = \{(s^+, p^+) \mid (\exists n, s^+, p^+).(s^+, p^+) \in b_{temp} : \\
&\qquad s^+ = do(n, s) \wedge choiceNat(n, a, s) \wedge PossAct(n, s) \wedge \\
&\qquad p^+ = p \cdot probObs(o, a, s^+) \cdot probNat(n, a, s)\} \\
&\quad b_{new} = normalize(b_{temp}).
\end{aligned}$$

$choiceNat(n, a, s)$ specifies the possible outcomes n of the agent's intention to perform action a. $PossAct(n, s)$ denotes the possibility of performing action n in situation s. $probObs(o, a, s^+)$ and $probNat(n, a, s)$ are functions that return the probability of observing o in the situation s^+—the situation resulting from doing action a, and respectively, the probability of action n being the outcome of the intention to execute action a in situation s.

For $BestDoPO$ to be integrated as required for the present work, two arguments are added to the list: $BestDoPO$ as defined in (Rens, Ferrein, and Van der Poel 2008) is modified to return δ and to take nom. The input program may provide information for a sequence of actions of length greater than the policy horizon. Call the remaining program δ—the portion of the program that was not used for policy generation. δ becomes the new program from which future policies will be generated. nom—the name of the input program—is used to select the reward function associated with the input program. Two clauses that are part of the definition of the modified $BestDoPO$ appear below.

$$
\begin{aligned}
&BestDoPO(p, \delta, nom, b, h, \pi, v, pr) \overset{def}{=}\\
&\quad h = 0 \wedge \delta = p \wedge\\
&\quad \pi = stop \wedge \exists v. believedReward(nom, v, b) \wedge pr = 1.\\
&BestDoPO(a : p, \delta, nom, b, h, \pi, v, pr) \overset{def}{=}\\
&\quad \neg actionBelievedPossible(a, b) \wedge\\
&\quad\quad \delta = p \wedge \pi = stop \wedge v = 0 \wedge pr = 0 \vee\\
&\quad actionBelievedPossible(a, b) \wedge\\
&\quad\quad \exists\, obs.setofAssocObservations(a, obs) \wedge\\
&\quad\quad \exists \pi', v', pr. Aux(obs, a, p, \delta, nom, b, h, \pi', v', pr) \wedge\\
&\quad\quad believedReward(nom, r, b) \wedge \pi = a; \pi' \wedge v = r + v'.
\end{aligned}
$$

Please refer to (Rens, Ferrein, and Van der Poel 2008) for more detail.

BDI Theory

A desire is understood as what an agent ideally wants to achieve, that is, what motivates it. In reality, agents are resource-bounded, and hence should rationally choose the desires to pursue whose achievement are most valuable to the agent and that are achievable according to the agent's current situation and capabilities. The desires that have been committed to pursuing through a rational process of reasoning may be called *intentions*. The Belief-Desire-Intention (BDI) model of agency takes intentions—in addition to beliefs and desires—as first-class mental states. Traditional agent architectures either simply do not consider intentions, or do not consider them as explicit operands within the processes of an agent's reasoning system.

The value of taking intentions seriously is that they manage the agent's resources in a rational way. Intentions induce the agent to act and intentions persist. As such, they focus the agent's activity to commit resources and thus pursue a desire more effectively. Also, because intentions persist, new intentions are not constantly being adopted: new intentions are constrained by current intentions, and hence, future deliberation is constrained (Wooldridge 2000).

It is useful to distinguish between *deliberation*: to decide on what ends (e.g., reward functions; goal states) to pursue and *means-ends reasoning*: how to achieve the ends. Deliberation may be further divided into (i) reasoning to generate options from beliefs, i.e., 'wishing' to decide on current desires; (ii) reasoning to select intentions, i.e., 'focusing' on a subset of those desires and committing to achieve them. Committed-to goals, or plans for achieving them, are *intentions*.

A BDI agent has at least these seven components (Wooldridge 1999):

- A knowledge base of beliefs.
- An option generation function ($wish$), generating the options the agent would ideally like to pursue (its desires).
- A set of desires $Dess$ returned by the $wish$ function.
- A function ($focus$) that filters out incompatible, impossible and less valuable desires, and that focuses on a subset of the desire set.
- A structure of intentions $Ints$—the most desireable options/desires returned by the $focus$ function.
- A belief change function ($update$): given the agent's current beliefs and the latest percept sensed, the belief change function returns the updated beliefs of the agent.
- A function ($execute$) that selects some action(s) from the plan the agent is currently executing, and executes the action(s).

In most of the well known implementations of agents based on the BDI model (e.g., PRS (Georgeff and Ingrand 1989), IRMA (Bratman, Israel, and Pollack 1988) and dMARS (Rao and Georgeff 1995)), the $plan$ function returns plans from a plan library; a set of pre-compiled plans. An *intention structure* then structures various plans into larger hierarchies of plans. An intention in the intention structure in the classical BDI theory is a partial plan structured as a hierarchy of subplans. Furthermore, subplans may at some point be abstract, waiting to be 'filled in' (Bratman, Israel, and Pollack 1988). Some BDI architectures are designed to let the $plan$ function generate plans from atomic actions (Sardina, De Silva, and Padgham 2006; Walczak et al. 2007) (or it may possibly use a combination of pre-compiled and generated plans). However, none of the architectures that have a generative component employ a planner that produces plans for a POMDP model.

Combining POMDP Planning with the BDI Model

In this section we see how an agent controller in the BDI model can incorporate the $BestDoPO$ POMDP planner into its practical reasoning processes. We took the prototypical control loop of the BDI model as a reference and modified it to accommodate planning with POMDP policies. The proposed architecture is called BDI-POP (BDI with POmdp Planner).

First we introduce some terms and their relationships with the aid of Figure 1 (next page). Implicitly included in the "BELIEF" data store, is a fixed set of behaviors $behs$ and a fixed set of reward functions $rwds$ ($rwds$ is considered globally accessible). $behs$ is the agent's primitive goals; its innate drive. The idea is that each behavior refers to a unique goal that the agent is designed to achieve. Each behavior is *defined* by the set of programs and reward functions that can achieve the behavior. The $wish$ function is omitted from our architecture (for now) because the options the agent would pursue at any time are its behaviors $beh s$. The agent also has a fixed set of desires d. Each $des \in d$ is a triple $(nom, prog, ach)$: nom is a reference to the Golog program $prog$, and ach is a reference to the behavior $beh \in behs$ that $prog$ can potentially achieve, thus $ach \in behs$. The reward functions $rf \in rwds$ take as argument a nom that refers to the program that rf is associated with. The following holds: $\forall beh.[beh \in behs \rightarrow (\exists des).des = (nom, prog, ach) \wedge ach = beh]$: for each behavior, there exists at least one program to achieve it.

To understand the controller, we also need to consider the agent's deliberation process. $deliberate$ is the procedure that calls and controls the $focus$ predicate and that operates on the intention stack. We write $focus(b, d, i, behs, h^-)$ to be the predicate that selects one $des \in d$ for each $beh \in behs$, placing these desires in a stack, in ascending order, ordered by the desires' values. The desire selected for a behavior is the one that can achieve the behavior ($ach = beh$) and that has the highest value. A desire's value is estimated as the value v of the policy found, generated to a depth h^-: $BestDoPO$ is called with b, h^- and the applicable $prog$ as arguments; v is used and the policy is discarded. We keep $h^- < h$ to save on time spent deliberating. $focus$ 'returns' the stack i of selected desires.

$$\begin{aligned}
&deliberate(b, d, i, behs, ai, i', h^-) \overset{def}{=}\\
&\quad (isEmpty(i) \wedge \exists i'.focus(b, d, i, behs, h^-) \vee\\
&\quad \neg isEmpty(i) \wedge \exists i'.i' = i) \wedge\\
&\quad \exists ai, i''.popIntentionStack(i', ai, i'').
\end{aligned}$$

BDI-POP tests whether a usable policy could be generated, that is, whether the planner returns the *stop* policy: When every outcome of an intended action (according to the input program) is illegal (according to the background action theory), $BestDoPO$ returns *stop*, and we say that the input program is *impossible*. An intention $i = (nom, prog, ach)$ with $prog$ being impossible is thus defined as an *impossible intention*.

The strategy used in $deliberate$ to deal with an impossible intention is extremely simple: it is dropped and the next intention on the stack is popped. This is a reasonable strategy because the next intention on the stack has the highest value, and should thus be pursued next. Calling $focus$ to refill the intention stack at this time would defeat the principle of *commitment* to intentions. Other strategies are possible, for example, replacing the impossible intention with another intention that achieves the same behavior, if one exists.

A logical high-level specification of BDI-POP follows, after which, it is explained in words.

$$\begin{aligned}
&Agent(b, d, i, behs, ai, \pi, h, h^-) \overset{def}{=}\\
&\quad (nom, p, ach) = ai \wedge\\
&\quad \pi \neq stop \wedge p \neq nil \wedge\\
&\quad \exists \pi''.\pi = a; \pi'' \wedge execute(a) \wedge\\
&\quad \exists sv.getPercep(a, sv) \wedge \exists o.recognize(a, o) \wedge\\
&\quad \exists \pi'''.getSubPolicy(\pi'', o, \pi''') \wedge\\
&\quad \exists b'.b' = BU(o, a, b)) \wedge\\
&\quad Agent(b', d, i, behs, ai, \pi''', h, h^-).
\end{aligned}$$

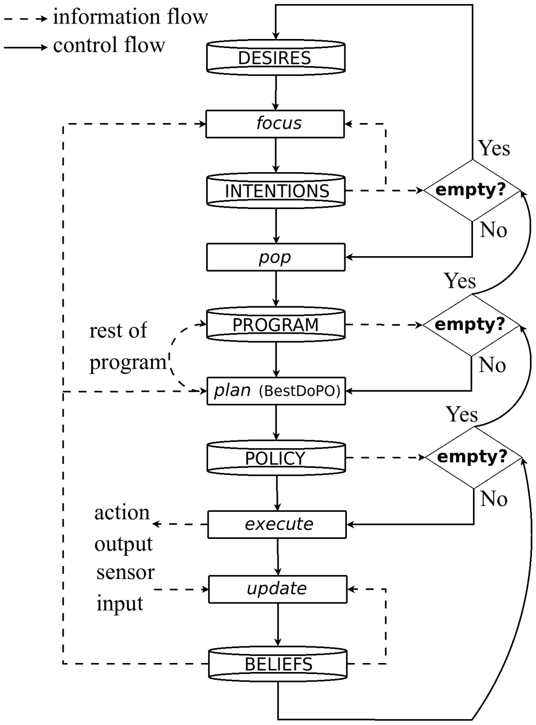

Figure 1: Schematic diagram of a sketch of the BDI architecture with the POMDP planner.

$$\begin{aligned}
&Agent(b, d, i, behs, ai, \pi, h, h^-) \overset{def}{=}\\
&\quad (nom, p, ach) = ai \wedge\\
&\quad \pi = stop \wedge p = nil \wedge\\
&\quad deliberate(b, d, i, behs, ai', i', h^-) \wedge\\
&\quad Agent(b, d, i', behs, ai', \pi, h, h^-).
\end{aligned}$$

$$\begin{aligned}
&Agent(b, d, i, behs, ai, \pi, h, h^-) \overset{def}{=}\\
&\quad (nom, p, ach) = ai \wedge\\
&\quad \pi = stop \wedge p \neq nil \wedge\\
&\quad \exists \delta, \pi', v, pr.BestDoPO(p, \delta, nom, b, h, \pi', v, pr) \wedge\\
&\quad ai' = (nom, \delta, ach) \wedge\\
&\quad (\pi' = stop \wedge\\
&\quad\quad \exists ai'', i'.deliberate(b, d, i, behs, ai'', i', h^-) \wedge\\
&\quad\quad Agent(b, d, i', behs, ai'', \pi', h, h^-) \vee\\
&\quad \pi' \neq stop \wedge\\
&\quad\quad \exists \pi''.\pi' = a; \pi'' \wedge execute(a) \wedge\\
&\quad\quad \exists sv.getPercep(a, sv) \wedge \exists o.recognize(a, o) \wedge\\
&\quad\quad \exists \pi'''.getSubPolicy(\pi'', o, \pi''') \wedge\\
&\quad\quad \exists b'.b' = BU(o, a, b)) \wedge\\
&\quad\quad Agent(b', d, i, behs, ai', \pi''', h, h^-)).
\end{aligned}$$

The agent follows the intention with the highest value—the intention popped from the stack. Call this the *active intention*. Initially, the intention stack is empty, so $deliberate$ is called and the active intention is instantiated. Whenever the controller needs a new plan to execute, $BestDoPO$ is called to generate a policy with horizon h using the program specified by the active intention. The agent executes the policy until the end of the policy is reached, then $BestDoPO$ is called again for the rest of the program. If there is no rest of program (the program is empty), $deliberate$ is called. If the program has become impossible, $deliberate$ is called.

$getPercept$ returns a sensor value, given the action executed / sensor activated. The agent processes the sensor data and decides what it observed—the agent recognizes the sensor reading via the $recognize$ predicate, which outputs an observation. With this observation, the correct subpolicy is extracted from the current policy, and this (possibly empty) subpolicy becomes the new current policy.

After the action recommended by the policy is executed, the agent's beliefs must be updated according to what it 'knows' about the effects of its actions. The same belief update function used during planning by $BestDoPO$ is used to update the agent's beliefs. The current belief state of the agent will be the 'initial' belief state required as argument to $BestDoPO$ the next time the planner is called.

Given our present definition of $deliberate$ and given that we shall allow only finite programs for achieving intentions, the agent is guaranteed to deliberate at regular intervals. However, this interval period is fixed (to the degree that intentions become impossible). Adding a $reconsider$ predicate that tells the agent once every control cycle whether to deliberate, is a more sophisticated method. $reconsider$ is described by, for example, Wooldridge (2000) and "was examined by David Kinny and Michael Georgeff, in a number of experiments," (Wooldridge 1999, p. 57). Because we are investigating the feasibility of the basic idea of the hybrid architecture in this paper, we have left out the $reconsider$ predicate from the present investigation.

A somewhat significant difference of our hybrid architecture from the perspective of control via POMDP policy generation, is that—as stand-alone controller—the POMDP planner takes a single plan with a single associated reward function, to generate a policy. The new hybrid architecture takes several programs, each with an associated reward function. This aspect of the agent being able to reason over multiple behaviors has the advantage that the agent designer can separately specify behaviors that should—at least intuitively—be considered separately.

$BestDoPO$ expands Golog programs into hierarchically structured plans (policies), and only programs that have been selected as intentions are expanded into policies. Each program can generate a policy—or several policies if the program is expanded piece-wise. Viewing a policy tree as an intention structure in the sense of traditional BDI architectures, each program in the intention stack represents (at least one) intention structure. BDI-POP, therefore, maintains *several* unexpanded intention structures, only expanded when popped from the intention stack.

Implementation and First Experiments

To validate the BDI-POP architecture and to gain a sense for its performance potential, we observe one agent based on the architecture, in a simulation. The simulation environment is inspired by Tileworld (Pollack and Ringuette 1990), a testbed for agents. We designed and implemented the FireEater world, a dynamically changing grid world (a 5 × 5, two dimensional grid of cells) in which our agent is situated. There are obstacles that change position and fires that can be 'eaten'. Space prohibits a detailed explanation of FireEater world.

The agent gets one 'fire-point' for eating one fire. It can only eat a fire if it is in the same cell as the fire. There are two agent behaviors: $\texttt{findFood}, \texttt{eat} \in behs$. `findFood` may be realized by two available programs, and `eat` is forced to be achieved by one (other) program. The agent can go left, right, up or down—locomotive actions which are stochastically nondeterministic; it can sense its location (probabilistically) and it can eat fire (deterministically).

In order to have a base-line against which the performance of the new hybrid architecture can be compared, a simple or 'naive' architecture (called Naive-POP) was implemented. It has no explicit intentions or desires as defined for the BDI model. The agent is provided with a single Golog program and associated reward function. In this implementation, the program loops continuously over a nondeterministic action—nondeterministic between all available actions. If there is no rest of program, that is, the agent has executed the whole program, the agent will stop its activity.

BDI-POP, in contrast, does not employ programs that loop infinitely (in the experiments): programs were designed so that they become empty as soon as a policy is generated from the program. Hence an intention will be regarded as 'achieved' as soon as its policy becomes empty. Then the next intention will be popped from the stack. Because the size of the stack equals $|behs|$ and because all programs are finite, it is guaranteed that periodically all intentions will be achieved, that is, the stack is empty, and the agent is forced to deliberate to fill the intention stack with a fresh set of intentions.

The two agents as implemented by the two architectures, have identical knowledge bases, except for their programs and reward functions. That is, they believe the same actions are possible, with the same effects and associated probabilities. They both employ the exact same planner: $BestDoPO$. The graph in Figure 2 compares the performance of the two architectures. Given the restrictions inherent in their respective architectures, each agent was roughly optimized to give them equal advantage.

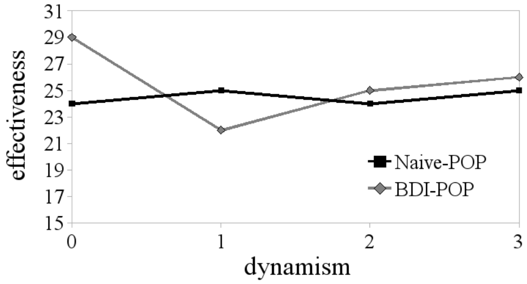

Figure 2: Performance of the two architectures.

Both architectures generate policies of horizon depth 3 ($h = 3$). In BDI-POP, we set $h^{-} = 1$—the search horizon that *focus* uses to determine program values. Throughout the experiments, the strategy for the time allowed to the agent was the same. There are 6 obstacles and initially 9 fires in each trial. We allow the agent to perform 3 actions each time before the obstacle positions change. The parameter for the number of obstacle changes per simulation cycle is the only parameter varied during experiments. Fourty trials per setting of this parameter were performed. *Effectiveness* is the total fire-points collected for the 40 trials. *Dynamism* is defined as 'number of obstacle changes per number of agent actions.

Conclusion

Compared to Naive-POP, the performance of BDI-POP in our experiments is not that impressive. This does not show that a BDI agent architecture should not be imbued with a POMDP planner; several enhancements to the simple BDI architecture used here are still possible: In particular, to build on this groundwork, we want to add the *reconsider* predicate to deal with cases where intentions have become inappropriate to some degree, and utilizing partial/abstract plan structures.

Moreover, the relative sophistication of BDI-POP may not be applicable in very simple worlds such as FireEater. We would thus like to deploy our agents in a larger world, perhaps with more complicated tasks for the agent to perform. This will also give more scope for the variety of programs that would be applicable, and the real power of the BDI model could come into play.

What *has* been shown is that the proposed architecture is implementable; there is no obvious fundamental conflict in synthesizing our POMDP planner and the BDI model for agent control. The groundwork has thus been laid for the development of more sophisticated planning processes in the BDI-POP framework.

What remains unclear is how practical this approach might be in realistically complex domains. With probabilistic outcomes and events in the world, the policy searches blow up very quickly with depth. For complete and optimal policies, POMDP solvers can deal with just a modest number of easily enumerated states. Policy trees of a fixed depth (as generated by *BestDoPO*) are not complete policies and thus less costly to generate. Realistically though, due to belief states being extremely numerous and the equivalence problem for states in the situation calculus, *BestDoPO* seems to be intractable if not undecidable. Furthermore, the situation calculus, in principle, provides a good deal of expressivity (including quantified reasoning), which brings its own computational complexity issues. For a hybrid BDI/POMDP architecture to scale up to a domain more meaningful than a microdomain, the integration of more 'common sense' reasoning techniques into the architecture may have benefits. And the latest advances in POMDP solvers (e.g., (Toussaint, Charlin, and Poupart 2008)) should be investigated for ideas to improve the efficiency of *BestDoPO*.

References

Bacchus, F.; Halpern, J.; and Levesque, H. 1999. Reasoning about noisy sensors and effectors in the situation calculus. *Artificial Intelligence* 1-2(111):171–208.

Boutilier, C.; Reiter, R.; Soutchanski, M.; and Thrun, S. 2000. Decision-theoretic, high-level agent programming in the situation calculus. In *Proc. AAAI-00*. AAAI Press. 355–362.

Bratman, M.; Israel, D.; and Pollack, M. 1988. Plans and resource-bounded practical reasoning. *Computational Intelligence* 4:349–355.

Bratman, M. 1987. *Intention, Plans, and Practical Reason*. Massachusetts/England: Harvard University Press.

Georgeff, M., and Ingrand, F. 1989. Decision-making in an embedded reasoning systems. In *Proc. IJCAI-89*. San Fransisco, CA: Morgan Kaufmann. 972–978.

Kaelbling, L.; Littman, M.; and Cassandra, A. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 1-2(101):99–134.

Levesque, H. J.; Reiter, R.; Lesperance, Y.; Lin, F.; and Scherl, R. B. 1997. GOLOG: A logic programming language for dynamic domains. *J. of Log. Progr.* 31(1-3).

Likhachev, M.; Ferguson, D.; Gordon, G.; Stentz, A.; and Thrun, S. 2005. Anytime dynamic A*: An anytime, replanning algorithm. In *Proc. Intl. Conf. on Automated Planning and Scheduling (ICAPS)*.

McCarthy, J. 1963. Situations, actions and causal laws. Technical report, Stanford University.

Pollack, M., and Ringuette, M. 1990. Introducing the Tileworld: Experimentally evaluating agent architectures. In *Proc. AAAI-90*. AAAI Press. 183–189.

Puterman, M. 1994. *Markov Decision Processes: Discrete Dynamic Programming*. New York, USA: Wiley.

Rao, A., and Georgeff, M. 1995. BDI agents: From theory to practice. In *Proc. ICMAS-95*. AAAI Press. 312–319.

Reiter, R. 2001. *Knowledge in Action*. MIT Press.

Rens, G.; Ferrein, A.; and Van der Poel, E. 2008. Extending DTGolog to deal with POMDPs. In *Proc. PRASA-08*. PRASA. 49–54.

Sardina, S.; De Silva, L.; and Padgham, L. 2006. Hierarchical planning in BDI agent programming languages: A formal approach. In *Proc. AAMAS-06*. ACM Press. 1001–1008.

Toussaint, M.; Charlin, L.; and Poupart, P. 2008. Hierarchical POMDP controller optimization by likelihood maximization. In *Workshop on Advancements in POMDP Solvers, Tech. Report WS-08-01, AAAI-08*. AAAI Press. url:http://www.aaai.org/Library/Workshops/ws08-01.php.

Walczak, A.; Braubach, L.; Pokahr, A.; and Lambersdorf., W. 2007. Augmenting BDI agents with deliberative planning techniques. In *Proc. ProMAS-06*. Springer. 113–127.

Wooldridge, M. 1999. Intelligent agents. In Weiss, G., ed., *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. Massachusetts/England: MIT Press. chapter 1.

Wooldridge, M. 2000. *Reasoning About Rational Agents*. Massachusetts/England: MIT Press.

HTN Planning with Preferences[*]

Shirin Sohrabi Jorge A. Baier Sheila A. McIlraith
Department of Computer Science
University of Toronto
{shirin, jabaier, sheila}@cs.toronto.edu

Abstract

In this paper we address the problem of generating preferred plans by combining the procedural control knowledge specified by Hierarchical Task Networks (HTNs) with rich user preferences. To this end, we extend the popular Planning Domain Definition Language, PDDL3, to support specification of simple and temporally extended preferences over HTN constructs. To compute preferred HTN plans, we propose a branch-and-bound algorithm, together with a set of heuristics that, leveraging HTN structure, measure progress towards satisfaction of preferences. Our preference-based planner, **HTNPLAN-P**, is implemented as an extension of the **SHOP2** planner. We compared our planner with **SGPlan**$_5$ and **HPLAN-P**– the top performers in the 2006 International Planning Competition preference tracks. **HTNPLAN-P** generated plans that in all but a few cases equalled or exceeded the quality of plans returned by **HPLAN-P** and **SGPlan**$_5$. While our implementation builds on **SHOP2**, the language and techniques proposed here are relevant to a broad range of HTN planners.

1 Introduction

Hierarchical Task Network (HTN) planning is a popular and widely used planning paradigm, and many domain-independent HTN planners exist (e.g., **SHOP2**, SIPE-2, I-X/I-PLAN, O-PLAN) [Ghallab *et al.*, 2004]. In HTN planning, the planner is provided with a set of tasks to be performed, possibly together with constraints on those tasks. A plan is then formulated by repeatedly decomposing tasks into smaller and smaller subtasks until primitive, executable tasks are reached. A primary reason behind HTN's success is that its task networks capture useful procedural control knowledge—advice on how to perform a task—described in terms of a decomposition of subtasks. Such control knowledge can significantly reduce the search space for a plan while also ensuring that plans follow one of the stipulated courses of action.

While HTNs specify a family of satisfactory plans, they are, for the most part, unable to distinguish between successful plans of differing quality. Preference-based planning (PBP) augments a planning problem with a specification of properties that constitute a high-quality plan. For example, if one were generating an air travel plan, a high-quality plan might be one that minimizes cost, uses only direct flights, and flies with a preferred carrier. PBP attempts to optimize the satisfaction of these preferences while achieving the stipulated goals of the plan. To develop a preference-based HTN planner, we must develop a specification language that references HTN constructs, and a planning algorithm that computes a preferred plan while respecting the HTN planning problem specification.

In this paper we extend the Planning Domain Definition Language, PDDL3 [Gerevini *et al.*, 2009], with HTN-specific preference constructs. This work builds on our recent work on the development of $\mathcal{LPH}$ [Sohrabi and McIlraith, 2008], a *qualitative* preference specification language designed to capture HTN-specific preferences. PDDL3 preferences are highly expressive, however they are solely *state centric*, identifying preferred states along the plan trajectory. To develop a preference language for HTN we add *action-centric* constructs to PDDL3 that can express preferences over the occurrence of primitive actions (operators) within the plan trajectory, as well as expressing preferences over complex actions (tasks) and how they decompose into primitive actions. For example, we are able to express preferences over which sets of subtasks are preferred in realizing a task (e.g., *When booking inter-city transportation, I prefer to book a flight*) and preferred parameters to use when choosing a set of subtasks to realize a task (e.g., *I prefer to book a flight with United*). To compute preferred HTN plans, we propose a branch-and-bound algorithm, together with a set of heuristics that leverage HTN structure.

The main contributions of this paper are: (1) a language that supports the specification of temporally extended preferences over complex action- and state-centric properties of a plan, and (2) heuristics and an algorithm that exploit HTN procedural preferences and control to generate preferred plans that under some circumstances are guaranteed optimal. The notion of adding advice to an HTN planner regarding how to decompose a task network was first proposed by Myers (e.g., [Myers, 2000]). Recently, there was another attempt to integrate preferences into HTN planning *without* the provision of action-centric language constructs [Lin *et al.*, 2008].

[*] This paper will also appear in the Proceedings of IJCAI-09.

We discuss these and other related works in Section 7. PBP has been the topic of much research in recent years, and there has been a resurgence of interest in HTN planning. Experimental evaluation of our planner shows that HTN PBP generates plans that, in all but a few cases, equal or exceed the best PBP planners in plan quality. As such, it argues for HTN PBP as a viable and promising approach to PBP.

2 Background

2.1 HTN Planning

Informally, an HTN planning problem can be viewed as a generalization of the classical planning paradigm. An HTN domain contains, besides regular primitive actions, a set of *tasks* or high-level actions. Tasks can be successively refined or *decomposed* by the application of so-called *methods*. When this happens, the task is replaced by a new, intuitively more specific *task network*. In short, a task network is a set of tasks plus a set of restrictions (often ordering constraints) that its tasks should satisfy. The HTN planning problem consists of finding a primitive decomposition of a given (initial) task network.

Example 1 (Travel Example) Consider the planning problem of arranging travel in which one has to arrange accommodation and various forms of transportation. This problem can be viewed as a simple HTN planning problem, in which there is a single task, “arrange travel”, which can be decomposed into arranging transportation, accommodations, and local transportation. Each of these more specific tasks can successively be decomposed based on alternative modes of transportation and accommodations, eventually reducing to primitive actions that can be executed in the world. Further constraints can be imposed to restrict decompositions.

A formal definition of HTN planning with preferences follows. Most of the basic definitions follow Ghallab *et al.* [2004].

Definition 1 (HTN Planning Problem) *An HTN planning problem is a 3-tuple $\mathcal{P} = (s_0, w_0, D)$ where s_0 is the initial state, w_0 is a task network called the initial task network, and D is the HTN planning domain which consists of a set of operators and methods.*

A domain is a pair $D = (O, M)$ where O is a set of operators and M is a set of methods. An operator is a primitive action, described by a triple *o =(name(o), pre(o), eff(o))*, corresponding to the operator’s name, preconditions and effects. In our example, ignoring the parameters, operators might include: *book-train*, *book-hotel*, and *book-flight*.

A *task* consists of a task symbol and a list of arguments. A task is primitive if its task symbol is an operator name and its parameters match, otherwise it is *nonprimitive*. In our example, *arrange-trans* and *arrange-acc* are nonprimitive tasks, while *book-flight* and *book-car* are primitive tasks.

A method, m, is a 4-tuple (*name(m), task(m),subtasks(m), constr(m))* corresponding to the method’s name, a nonprimitive task and the method’s task network, comprising subtasks and constraints. Method m is relevant for a task t if there is a substitution σ such that $\sigma(t)$ =*task*(m). Several methods can be relevant to a particular nonprimitive task t, leading to different decompositions of t. In our example, the method with *name by-flight-trans* can be used to decompose the *task arrange-trans* into the *subtasks* of booking a flight and paying, with the constraint (*constr*) that the booking precede payment. An operator o may also accomplish a ground primitive task t if their names match.

Definition 2 (Task Network) *A task network is a pair w=(U, C) where U is a set of task nodes and C is a set of constraints. Each task node u ∈ U contains a task t_u. If all of the tasks are primitive, then w is called primitive; otherwise it is called nonprimitive.*

In our example, we could have a task network (U, C) where $U = \{u_1, u_2\}$, u_1 =*book-car*, and u_2= *pay*, and C is a precedence constraint such that u_1 must occur before u_2 and a before-constraint such that at least one car is available for rent before u_1.

Definition 3 (Plan) $\pi = o_1 o_2 \ldots o_k$ *is a plan for HTN planning program* $\mathcal{P} = (s_0, w_0, D)$ *if there is a primitive decomposition,* w*, of* w_0 *of which* π *is an instance.*

Finally, to define the notion of *preference-based* planning we assume the existence of a reflexive and transitive relation $\preceq$ between plans. If π_1 and π_2 are plans for $\mathcal{P}$ and $\pi_1 \preceq \pi_2$ we say that π_1 is *at least as preferred as* π_2. We use $\pi_1 \prec \pi_2$ as an abbreviation for $\pi_1 \preceq \pi_2$ and $\pi_2 \not\preceq \pi_1$.

Definition 4 (Preference-based HTN Planning) *An HTN planning problem with user preferences is described as a 4-tuple* $\mathcal{P} = (s_0, w_0, D, \preceq)$ *where* $\preceq$ *is a preorder between plans. A plan* π *is a solution to* $\mathcal{P}$ *if and only if:* π *is a plan for* $\mathcal{P}' = (s_0, w_0, D)$ *and there does not exists a plan* π' *for* $\mathcal{P}'$ *such that* $\pi' \prec \pi$.

The $\preceq$ relation can be defined in many ways. Below we describe PDDL3, which defines $\preceq$ quantitatively through a metric function.

2.2 Brief Description of PDDL3

The Planning Domain Definition Language (PDDL) is the de facto standard input language for many planning systems. PDDL3 [Gerevini *et al.*, 2009] extends PDDL2.2 to support the specification of preferences and hard constraints over *state* properties of a trajectory. These preferences form the building blocks for definition of a PDDL3 *metric function* that defines the quality of a plan. In this context PBP necessitates maximization (or minimization) of the metric function. In what follows, we describe those elements of PDDL3 that are most relevant to our work.

Temporally extended preferences/constraints PDDL3 specifies temporally extended preferences (TEPs) and temporally extended hard constraints in a subset of linear temporal logic (LTL). Preferences are given names in their declaration, to allow for later reference. The following PDDL3 code illustrates one preference and one hard constraint.

```
(forall (?l - light)
 (preference p-light (sometime (turn-off ?l))))
(always (forall ?x - explosive)
                   (not (holding ?x)))
```

The `p-light` preference suggests that the agent eventually turn all the lights off. The (unnamed) hard constraint establishes that an explosive object cannot be held by the agent at any point in a valid plan.

When a preference is *externally* universally quantified, it defines a family of preferences, comprising an individual preference for each binding of the variables in the quantifier. Therefore, preference `p-light` defines an individual preference for each object of type `light` in the domain.

Temporal operators cannot be nested in PDDL3. Our approach can however handle the more general case of nested temporal operators.

Precondition Preferences Precondition preferences are atemporal formulae expressing conditions that should ideally hold in the state in which the action is performed. They are defined as part of the action's precondition.

Simple Preferences Simple preferences are atemporal formulae that express a preference for certain conditions to hold in the final state of the plan. They are declared as part of the goal. For example, the following PDDL3 code:

```
(:goal
   (and (delivered pkg1 depot1)
        (preference p-truck (at truck depot1))))
```

specifies both a hard goal (`pkg1` must be delivered at `depot1`) and a simple preference (that `truck` is at `depot1`). Simple preferences can also be quantified.

Metric Function The metric function defines the quality of a plan, generally depending on the preferences that have been achieved by the plan. To this end, the PDDL3 expression `(is-violated name)`, returns the number of individual preferences in the `name` family of preferences that have been violated by the plan.

Finally, it is also possible to define whether we want to maximize or minimize the metric, and how we want to weigh its different components. For example, the PDDL3 metric function:

```
(:metric minimize (+
                   (* 40 (is-violated p-light))
                   (* 20 (is-violated p-truck))))
```

specifies that it is twice as important to satisfy preference `p-light` as to satisfy preference `p-truck`.

Since it is always possible to transform a metric that requires maximization into one that requires minimization, we will assume that the metric is always being *minimized*.

Finally, we now complete the formal definition for HTN planning with PDDL3 preferences. Given a PDDL3 metric function M the *HTN preference-based planning problem with PDDL3 preferences* is defined by Definition 4, where the relation $\preceq$ is such that $\pi_1 \preceq \pi_2$ iff $M(\pi_1) \leq M(\pi_2)$.

3 PDDL3 Extended to HTN

In this section, we extend PDDL3 with the ability to express preferences over HTN constructs. As argued in Section 1, supporting preferences over how tasks are decomposed, their preferred parameterizations, and the conditions underwhich these preferences hold, is compelling. It goes beyond the traditional specification of preferences over the properties of states within plan trajectories to provide preferences over non-functional properties of the planning problem including *how* some planning objective is accomplished. This is particularly useful when HTN methods are realized using web service software components, because these services have many non-functional properties that distinguish them (e.g., credit cards accepted, country of origin, trustworthiness, etc.) and that influence user preferences.

In designing a preference specification language for HTN planning, we made a number of strategic design decisions. We first considered adding our preference specifications directly to the definitions of HTN methods. This seemed like a natural extension to the hard constraints that are already part of method definitions. Unfortunately, this precludes easy contextualization of methods relative to the task the method is realizing. For example, in the travel domain, many methods may eventually involve the primitive operation of *paying*, but a user may prefer different methods of payment dependent upon the high-level task being realized (e.g., *When booking a car, pay with amex to exploit amex's free collision coverage, when booking a flight, pay with my Aeroplan-visa to collect travel bonus points*, etc.). We also found the option of including preferences in method definitions unappealing because we wished to separate domain-specific, but user-independent knowledge, such as method definitions, from user-specific preferences. Separating the two, enables users to share method definitions but individualize preferences. We also wished to leverage the popularity of PDDL3 as a language for preference specifications.

Here, we extend PDDL3 to incorporate complex action-centric preferences over HTN tasks. This gives users the ability to express preferences over certain parameterization of a task (e.g., preferring one task grounding to another) and over certain decompositions of nonprimitive tasks (i.e., prefer to apply a certain method over another). To support preferences over task occurrences (primitive and nonprimitive) and task decompositions, we added three new constructs to PDDL3: **occ**(a), **initiate**(x) and **terminate**(x), where a is a primitive task (i.e., an action), and x is either a task or a name of method. **occ**(a) states that the primitive task a occurs in the present state. On the other hand **initiate**(t) and **terminate**(t) state, respectively, that the task t is initiated or terminated in the current state. Similarly **initiate**(n) (resp. **terminate**(n)) states that the application of method named n is initiated (resp. terminated) in the current state. These new constructs can be used within simple and temporally extended preferences and constraints, but not within precondition preferences.

The following are a few temporally extended preferences from our travel domain[1] that use the above extension.

```
(preference p1
  (always (not (occ (pay MasterCard)))))
(preference p2 (sometime (occ
  (book-flight SA Eco Direct WindowSeat))))
(preference p3 (imply (close origin dest)
  (sometime (initiate (by-rail-trans)))))
(preference p4
  (sometime-after (terminate (arrange-trans))
                  (initiate (arrange-acc))))
```

The **p1** preference states that the user never pays by Mastercard. The **p2** preference states that at some point the user

[1] For simplicity many parameters have been suppressed.

books a direct economy window-seated flight with a Star Alliance (SA) carrier. The **p3** preference states that the *by-rail-trans* method is applied when origin is close to destination. Finally **p4** states that *arrange-trans* task is terminated before the *arrange-acc* task begins (for example: finish arranging your transportation before booking a hotel).

Semantics: The semantics of the preference language comprises two parts: (1) a formal definition of the satisfaction of individual preference formulae, and (2) a formal definition of the aggregation of preferences through an objective function. The satisfaction of individual preference formulae is defined by mapping HTN decompositions and LTL formulae into the situation calculus [Reiter, 2001]. In so doing, satisfaction of a preference formula is reduced to entailment of the formula in a logical theory. A sketch of the situation calculus encoding is found in Appendix A. Preference formulae are composed into a metric function. The semantics of the metric function, including the aggregation of quantified preferences via the `is-violated` function, is defined in the same way as in PDDL3, following Gerevini *et al.* [2009].

4 Preprocessing HTN problems

Before searching for a most preferred plan, we preprocess the original problem. This is needed in order to make the planning problem more easily manageable by standard planning techniques. We accomplish this objective by removing all of the modal operators appearing in the preferences. The resulting domain, has only final-state preferences, and all preferences refer to state properties.

By converting TEPs into final-state preferences, our heuristic functions are only defined in terms of domain predicates, rather than being based on non-standard evaluations of an LTL formula, such as the ones used by other approaches [e.g. Bienvenu *et al.*, 2006]. Nor do we need to implement specialized algorithms to reason about LTL formulae such as the progression algorithm used by **TLPLAN** [Bacchus and Kabanza, 1998].

Further, by removing the modal operators **occ**, **initiate**, and **terminate** we provide a way to refer to these operators via state predicates. This allows us to use standard HTN planning software as modules of our planner, without needing special modifications such as a mechanism to keep track of the tasks that have been decomposed or the methods that have been applied.

Preprocessing Tasks and Methods Our preferences can refer to the occurrence of tasks and the application of methods. In order to reason about task occurrences and method applications, we preprocess the methods of our HTN problem. In the compiled problem, for each non-primitive task t that occurs in some preference of the original problem, there are two new predicates: $executing\text{-}t$ and $terminated\text{-}t$. If $a_0 a_1 \cdots a_n$ is a plan for the problem, and a_i and a_j are respectively the first and last primitive actions that resulted from decomposing t, then $executing\text{-}t$ is true in all the states in between the application of a_i and a_j, and $terminated\text{-}t$ is true in all states after a_j. This is accomplished by adding new actions at the beginning and end of each task network in the methods that decompose t. Further, for each primitive task (i.e., operator) t occurring in the preferences, we extend the compiled problem with a new $occ\text{-}t$ predicate, such that $occ\text{-}t$ is true iff t has just been performed.

Finally, we modify each method m whose name n (i.e., $n = name(m)$) that occurs in some preference. We use two predicates $executing\text{-}n$ and $terminated\text{-}n$, whose updates are realized analogously to their task versions described above.

Preprocessing the Modal Operators We replace each occurrence of **occ**(t), **initiate**(t), and **terminate**(t) by $occ\text{-}t$ when t is primitive. We replace the occurrence of **initiate**(t) by $executing\text{-}t$, and **terminate**(t) by $terminated\text{-}t$ when t is non-primitive. Occurrences of **initiate**(n) are replaced by $executing\text{-}n$, and **terminate**(n) by $terminated\text{-}n$.

Up to this point all our preferences exclusively reference predicates of the HTN problem, enabling us to apply standard techniques to simplify the problem further.

Temporally Extended and Precondition Preferences We use an existing compilation technique [Baier *et al.*, 2009] to encode the satisfaction of temporally extended preferences into predicates of the domain. For each LTL preference φ in the original problem, we generate additional predicates for the compiled domain that encode the various ways in which φ can become true. Indeed, the additional predicates represent a finite-state automaton for φ, where the accepting state of the automaton represents satisfaction of the preference. In our resulting domains, we axiomatically define an *accepting predicate for* φ, which represents the accepting condition of φ's automaton. The accepting predicate is true at a state s if and only if φ is satisfied at s. Quantified preferences are compiled into parametric automata for efficiency. Finally, precondition preferences, preferences that should ideally hold in the state in which the action is performed, are compiled away as conditional action costs, as is done in the **HPLAN-P** planner. For more details refer to the original paper [Baier *et al.*, 2009].

5 Preference-based Planning with HTNs

We address the problem of finding a most preferred decomposition of an HTN by performing a best-first, incremental search in the plan search space induced by the initial task network. The search is performed in a series of *episodes*, each of which returns a sequence of ground primitive operators (i.e., a plan that satisfies the initial task network). During each episode, the search performs branch-and-bound pruning—a search node is pruned from the search space, if we can prove that it will not lead to a plan that is better than the one found in the previous episode. In the first episode no pruning is performed. In each episode, search is guided by *inadmissible heuristics*, designed specifically to guide the search quickly to a good decomposition. The remainder of this section describes the heuristics we use, and the planning algorithm.

5.1 Algorithm

Our HTN PBP algorithm outlined in Figure 1, performs a best-first, incremental search in the space of decompositions of a given initial task network. It takes as input a planning problem (s_0, w_0, D), a metric function METRICFN, and a heuristic function HEURISTICFN.

```
 1: function HTNPBP($s_0$, $w_0$,$D$, METRICFN,HEURISTICFN)
 2:   $frontier \leftarrow \langle s_0, w_0, \emptyset\rangle$          ▷ initialize frontier
 3:   $bestMetric \leftarrow$ worst case upper bound
 4:   while $frontier$ is not empty do
 5:     $current \leftarrow$ Extract best element from $frontier$
 6:     $\langle s, w, partialP\rangle \leftarrow current$
 7:     $lbound \leftarrow$ METRICBOUNDFN($s$)
 8:     if $lbound < bestMetric$ then          ▷ pruning by bounding
 9:       if $w = \emptyset$ and $current$'s metric $< bestMetric$ then
10:         Output plan $partialP$
11:         $bestMetric \leftarrow$ METRICFN($s$)
12:       $succ \leftarrow$ successors of $current$
13:       $frontier \leftarrow$ merge $succ$ into $frontier$
```

Figure 1: A sketch of our HTN PBP algorithm.

The main variables kept by the algorithm are *frontier* and $bestMetric$. $frontier$ contains the nodes in the search frontier. Each of these nodes is of the form $\langle s, w, partialP\rangle$, where s is a plan state, w is a task network, and $partialP$ is a partial plan. Intuitively, a search node $\langle s, w, partialP\rangle$ represents the fact that task network w remains to be decomposed in state s, and that state s is reached from the initial state of the planning problem s_0 by performing the sequence of actions $partialP$. $frontier$ is initialized with a single node $\langle s_0, w_0, \emptyset\rangle$, where $\emptyset$ represents the empty plan. Its elements are always sorted according to the function HEURISTICFN. On the other hand, $bestMetric$ is a variable that stores the metric value of the best plan found so far, and it is initialized to a high value representing a worst case upper bound.

Search is carried out in the main **while** loop. In each iteration, **HTNPLAN-P** extracts the best element from the $frontier$ and places it in $current$. Then, an estimation of a lowerbound of the metric value that can be achieved by decomposing w – $current$'s task network – is computed (Line 7) using the function METRICBOUNDFN. Function METRICBOUNDFN will be computed using the *optimistic metric* function described in the next subsection.

The algorithm *prunes* $current$ from the search space if $lbound$ is greater than or equal to $bestMetric$. Otherwise, **HTNPLAN-P** checks whether or not $current$ corresponds to a plan (this happens when its task network is empty). If $current$ corresponds to a plan, the sequence of actions in its tuple is returned and the value of $bestMetric$ is updated.

Finally, all successors to $current$ are computed using the Partial-order Forward Decomposition procedure (PFD) [Ghallab *et al.*, 2004], and merged into the frontier. The algorithm terminates when $frontier$ is empty.

5.2 Heuristics

Our algorithm searches for a plan in the space of all possible decompositions of the initial task network. HTNs that have been designed specifically to be customizable by user preferences may contain tasks that could be decomposed by a fairly large number of methods. In this scenario, it is essential for the algorithm to be able to evaluate which methods to use to decompose a task in order to get to a reasonably good solution quickly. The heuristics we propose in this section are specifically designed to address this problem. All heuristics are evaluated in a search node $\langle s, w, partialP\rangle$.

Optimistic Metric Function (OM**)** This function is an estimate of the best metric value achievable by any plan that can result from the decomposition of the current task network w. Its value is computed by evaluating the metric function in s but assuming that (1) no further precondition preferences will be violated in the future, (2) temporally extended preference that are violated and that can be proved to be unachievable from s are regarded as false, (3) all remaining preferences are regarded as satisfied. To prove that a temporally extended preference p is unachievable from s, OM uses a sufficient condition: it checks whether or not the automaton for p is currently in a state from which there is no path to an accepting state. Recall that an accepting state is reached when the preference formula is satisfied.

OM provides a lower bound on the best plan extending the partial plan $partialP$ assuming that the metric function is non-decreasing in the number of violated preferences. This is the function used as METRICBOUNDFN in our planner. OM is a variant of "optimistic weight" [Bienvenu *et al.*, 2006].

Pessimistic Metric Function (PM**)** This function is the dual of OM. While OM regards anything that is not provably violated (regardless of future actions) as satisfied, PM regards anything that is not provably satisfied (regardless of future actions) as violated. Its value is computed by evaluating the metric function in s but assuming that (1) no further precondition preferences will be violated in the future, (2) temporally extended preferences that are satisfied and that can be proved to be true in any successor of s are regarded as satisfied, (3) all remaining preferences are regarded as violated. To prove that a temporally extended preference p is true in any successor of s, we check whether in the current state of the world the automaton for p would be in an accepting state that is also a sink state, i.e., from which it is not possible to escape, regardless of the actions performed in the future.

For reasonable metric functions (e.g., those non-decreasing in the number of violated preferences), PM is monotonically decreasing as more actions are added to $partialP$. PM provides good guidance because it is a measure of assured progress towards the satisfaction of the preferences.

Lookahead Metric Function (LA**)** This function is an estimate of the metric of the *best successor* to the current node. It is computed by conducting a two-phase search. In the first phase, a search for all possible decompositions of w is performed, up to a certain depth k. In the second phase, for each of the resulting nodes, a single primitive decomposition is computed, using depth-first search. The result of LA is the best metric value among all the fully decomposed nodes. Intuitively, LA estimates the metric value of a node by first performing an exhaustive search for decompositions of the current node, and then by approximating the metric value of the resulting nodes by the metric value of the the first primitive decomposition that can be found, a form of sampling of the remainder of the search space.

Depth (D**)** We use the depth as another heuristic to guide the search. This heuristic does not take into account the preferences. Rather, it encourages the planner to find a decomposition soon. Since the search is guided by the HTN structure, guiding the search toward finding a plan using depth is natural. Other HTN planners such as **SHOP2** also use depth or depth-first search to guide the search to find a plan quickly.

Strategy	Check whether	If tied	If tied
No-LA	$OM_1 < OM_2$	$PM_1 < PM_2$	-
LA	$LA_1 < LA_2$	$OM_1 < OM_2$	$PM_1 < PM_2$

Figure 2: Strategies to determine whether a node n_1 is better than a node n_2. OM is the optimistic-metric, PM is the pessimistic-metric, and LA is the look-ahead heuristic.

The HeuristicFn function we use in our algorithm corresponds to a *prioritized sequence* of the above heuristics, in which D is always considered first. As such, when comparing two nodes we look at their depths, returning the one that has a higher depth value. If the depths are equal, we use the other heuristics in sequence to break ties. Figure 2 outlines the sequences we have used in our experiments.

5.3 Optimality and Pruning

Since we are using inadmissible heuristics, we cannot guarantee that the plans we generate are optimal. The only way to do this is to run our algorithm until the space is exhausted. In this case, the final plan returned is guaranteed to be optimal.

Exhaustively searching the search space is not reasonable in most planning domains, however here we are able to exploit properties of our planning problem to make this achievable some of the time. Specifically, most HTN specifications severely restrict the search space so that, relative to a classical planning problem, the search space is exhaustively searchable. Further, in the case where our preference metric function is additive, our OM heuristic function enables us to soundly prune partial plans from our search space. Specifically, we say that a pruning strategy is sound if and only if whenever a node is pruned (line 8) the metric value of any plan extending this node will exceed the current bound $bestMetric$. This means that no state will be incorrectly pruned from the search space.

Proposition 1 *The OM function provides sounds pruning if the metric function is non-decreasing in the number of satisfied preferences, non-decreasing in plan length, and independent of other state properties.*

A metric is non-decreasing in plan length if one cannot make a plan better by increasing its length only (without satisfying additional preferences).

Theorem 1 *If the algorithm performs sound pruning, then the last plan returned, if any, is optimal.*

Proof sketch: Follows the proof of optimality for the **HPlan-P** planner [Baier *et al.*, 2009].

6 Implementation and Evaluation

Our implemented HTN PBP planner, **HTNPlan-P**, has two modules: a preprocessor and a preference-based HTN planner. The preprocessor reads PDDL3 problems and generates a **SHOP2** planning problem with only simple (final-state) preferences. The planner itself is a modification of the LISP version of **SHOP2** [Nau *et al.*, 2003] that implements the algorithm and heuristics described above.

We had three objectives in performing our experimental evaluation: to evaluate the relative effectiveness of our heuristics, to compare our planner with state-of-the-art PBP planners, and to compare our planner with other HTN PBP planners. Unfortunately, we were unable to achieve our third objective, since we could not obtain a copy of **SCUP**, the only HTN PBP planner we know of [Lin *et al.*, 2008]. (See Section 7 for a qualitative comparison.)

		HTNPlan-P							
		No-LA		*LA*		**SGPlan$_5$**		**HPlan-P**	
	#Prb	#S	#Best	#S	#Best	#S	#Best	#S	#Best
travel	41	41	3	41	**37**	41	1	41	17
rovers	20	20	4	20	**19**	20	1	11	2
trucks	20	20	6	20	**15**	20	11	4	2

Figure 3: Comparison between two configurations of **HTNPlan-P**, **HPlan-P**, and **SGPlan$_5$** on **rovers**, **trucks**, and **travel** domains. Entries show number of problems in each domain (#Prb), number of solved instances in each domain (#S) by each planner, and number of times each planner found a plan of equal or better quality to those found by all other planners (#Best). All planners were ran for 60 minutes, and with a limit of 2GB per process.

We used three domains for the evaluation: the **rovers** domain, the **trucks** domain, both standard IPC benchmark domains; and the **travel** domain, which is a domain of our own making. Both the **rovers** and **trucks** domains comprised the preferences from IPC-5. In **rovers** domain we used the HTN designed by the developers of **SHOP2** for IPC-2 and in **trucks** we created our own HTN. We modified the HTN in **rovers** very slightly to reflect the true nondeterminism in our **HTNPlan-P** planner: i.e., if a task could be decomposed using two different methods, then both methods would be considered, not just the first applicable one. We also modified the IPC-5 preferences slightly to ensure fair comparison between planners. The **rovers** and **trucks** problems sets comprised 20 problems. The number of preferences in these problem sets ranged in size, with several having over 100 preferences per problem instance.

The **travel** domain is a PDDL3 formulation of the domain introduced in Example 1. Its problem set was designed in order to evaluate the PBP approaches based on two dimensions: (1) scalability, which we achieved by increasing the branching factor and grounding options of the domain, and (2) the complexity of the preferences, which we achieved by injecting inconsistencies (i.e., conflicts) among the preferences. In particular, we created 41 problems with preferences generated automatically with increasing complexity. For example problem 3 has 27 preferences with 8 conflicts in the choice of transportation while problem 40 has 134 preferences with 54 conflicts in the choice of transportation.

Our experiments evaluated the performance of four planners: **HTNPlan-P** with the *No-LA* heuristic, and **HTNPlan-P** with the *LA* heuristic, **SGPlan$_5$** [Hsu *et al.*, 2007], and **HPlan-P**– the latter two being the top PBP performers at IPC-5. Results are summarized in Figure 3, and show that **HTNPlan-P** generated plans that in all but a few cases equalled or exceeded the quality of plans returned by **HPlan-P** and **SGPlan$_5$**. The results also show that **HTNPlan-P** performs better on the three domains with the *LA* heuristic.

Conducting the search in a series of episodes does help in finding better-quality plans. To evaluate this, we calculated the *percent metric improvement* (PMI), i.e., the percent difference between the metric of the first and the last plan returned by our planner (relative to the first plan). The average PMI is

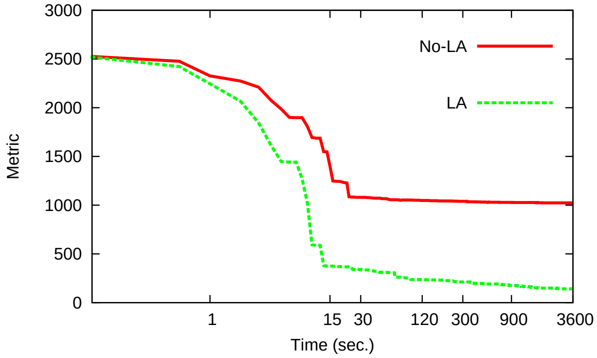

Figure 4: Added metric vs. time for the two strategies in the trucks domain. Recall that a low metric value means higher quality plan. When a problem is not solved at time t, we add its worst possible metric value (i.e. we assume no satisfied preferences).

40% in **rovers**, 72% in **trucks**, and 8% in **travel**.

To compare the relative performance between *LA* and *No-LA*, we averaged the percent metric difference (relative to the worst plan) in problems in which the configurations found a different plan. This difference is 45% in **rovers**, 60% in **trucks**, and 3% in **travel**, all in favour of *LA*.

We created 18 instances of the **travel** domain where we tested the performance between *LA* and *No-LA* on problems that have preferences that use our HTN extension of PDDL3. The average PMI for these problems is 13%, and the relative performance between the two is 5%.

Finally Figure 4 shows the decrease of the sum of the metric value of all instances of the trucks domain relative to solving time. We observe a rapid improvement during the first seconds of search, followed by a marginal one after 900 seconds. Other domains exhibit similar behaviour.

7 Discussion and Related Work

PBP has been the subject of much interest recently, spurred on by three IPC-5 tracks on this subject. A number of planners were developed, all based on the competition's PDDL3 language. Our work is distinguished in that it employs HTN domain control extending PDDL3 with HTN-inspired constructs. The planner itself then employs heuristics and algorithms that exploit HTN-specific preferences and control. Experimental evaluation of our planner shows that **HTNPLAN-P** generates plans that, in all but a few cases, equal or exceed the best PBP in plan quality. As such, it argues generally for HTN PBP as a viable and promising approach to PBP.

With respect to advisable HTN planners, Myers was the first to advocate augmenting HTN planning with hard constraints to capture advice on *how to* decompose HTNs, extending the approach to conflicing advice in [Myers, 2000]. Their work is similar in vision and spirit to our work, but different with respect to the realization. In their work, preferences are limited to consistent combinations of HTN advise; they do no include the rich temporally extended state-centric preferences found in PDDL3, nor do they support the weighted combination of preferences into a metric function that defines plan quality. With respect to computing HTN PBP, Myers' algorithm does not exploit lookahead heuristics or sound pruning techniques.

The most notable related work is that of Lin *et al.* [2008] who developed a prototype HTN PBP planner, **SCUP**, tailored to the task of web service composition. Unfortunately, **SCUP** is not available for experimental comparison, however there are fundamental differences between the planners, that limit the value of such a comparison. Most notably, Lin *et al.* [2008] do not extend PDDL3 with HTN-specific preference constructs, a hallmark of our work. Further, their planning algorithm appears to be unable to handle conflicting user preferences since they note that such conflict detection is performed manually prior to invocation of their planner. Optimization of conflicting preferences is common in most PBP's, including ours. Also, their approach to HTN PBP planning is quite different from ours. In particular, they translate user preferences into HTN constraints and preprocess the preferences to check if additional tasks need to be added to w_0. This is well motivated by the task of web service composition, but not a practice found in classical HTN planning.

Also related is the **ASPEN** planner [Rabideau *et al.*, 2000], which performs a simple form of preference-based planning, focused mainly on preferences over resources. It can plan with HTN-like task decomposition, but its preference language is far less expressive than ours. In contrast to **HTNPLAN-P**, **ASPEN** performs local search for a local optimum. It does not perform well when preferences are interacting, nested, or not local to a specific activity.

It is interesting and important to note that the HTN planners **SHOP2** [Nau *et al.*, 2003] and **ENQUIRER** [Kuter *et al.*, 2004] can be seen to handle some simple user preferences. In particular the order of methods and sorted preconditions in a domain description specifies a user preference over which method is more preferred to decompose a task. Hence users may write different versions of a domain description to specify simple preferences. However, unlike **HTNPLAN-P** the user constraints are treated as hard constraints and (partial) plans that do not meet these constraints will be pruned from the search space.

Finally, observe that we approached HTN PBP by integrating PBP into HTN planning. An alternative approach would be to integrate HTN into PBP. Kambhampati *et al.* [1998] hints at how this might be done by integrating HTN into their plan repair planning paradigm. For the integration of HTN into PBP to be effective, heuristics would have to be developed that exploited the special compiled HTN structure. Further, such a compilation would not so easily lend itself to mixed-initiative PBP, a topic for future investigation.

Acknowledgements: We thank our colleague Christian Fritz for helpful discussion. We gratefully acknowledge funding from the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Ontario Ministry of Innovations Early Researcher Award (ERA).

References

[Bacchus and Kabanza, 1998] F. Bacchus and F. Kabanza. Planning for temporally extended goals. *Annals of Mathematics and Artificial Intelligence*, 22(1-2):5–27, 1998.

[Baier *et al.*, 2009] J. A. Baier, F. Bacchus, and S. A. McIlraith. A heuristic search approach to planning with temporally extended preferences. *Artificial Intelligence*, 173(5-6):593–618, 2009.

[Bienvenu *et al.*, 2006] M. Bienvenu, C. Fritz, and S. A. McIlraith. Planning with qualitative temporal preferences. In *Proc. of the 10th Int'l Conference on Knowledge Representation and Reasoning (KR)*, 134–144, 2006.

[Gabaldon, 2002] A. Gabaldon. Programming hierarchical task networks in the situation calculus. In *AIPS'02 Workshop on On-line Planning and Scheduling*, April 2002.

[Gabaldon, 2004] A. Gabaldon. Precondition control and the progression algorithm. In *Proc. of the 9th Int'l Conference on Knowledge Representation and Reasoning (KR)*, 634–643. AAAI Press, 2004.

[Gerevini *et al.*, 2009] A. Gerevini, P. Haslum, D. Long, A. Saetti, and Y. Dimopoulos. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence*, 173(5-6):619–668, 2009.

[Ghallab *et al.*, 2004] M. Ghallab, D. Nau, and P. Traverso. *Hierarchical Task Network Planning. Automated Planning: Theory and Practice*. Morgan Kaufmann, 2004.

[Hsu *et al.*, 2007] C.-W. Hsu, B. Wah, R. Huang, and Y. Chen. Constraint partitioning for solving planning problems with trajectory constraints and goal preferences. In *Proc. of the 20th Int'l Joint Conference on Artificial Intelligence (IJCAI)*, 1924–1929, 2007.

[Kambhampati *et al.*, 1998] S. Kambhampati, A. D. Mali, and B. Srivastava. Hybrid planning for partially hierarchical domains. In *Proc. of the 15th National Conference on Artificial Intelligence (AAAI)*, 882–888, 1998.

[Kuter *et al.*, 2004] U. Kuter, E. Sirin, D. S. Nau, B. Parsia, and J. A. Hendler. Information gathering during planning for web service composition. In *Proc. of the 3rd Int'l Semantic Web Conference (ISWC)*, 335–349, 2004.

[Lin *et al.*, 2008] N. Lin, U. Kuter, and E. Sirin. Web service composition with user preferences. In *Proceedings of the 5th European Semantic Web Conference (ESWC)*, 629–643, 2008.

[Myers, 2000] K. L. Myers. Planning with conflicting advice. In *Proc. of the 5th Int'l Conference on Artificial Intelligence Planning and Scheduling (AIPS)*, 355–362, 2000.

[Nau *et al.*, 2003] D. Nau, T.-C. Au, O. Ilghami, U. Kuter, J. Murdock, D. Wu, and F. Yaman. SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research*, 20:379–404, 2003.

[Rabideau *et al.*, 2000] G. Rabideau, B. Engelhardt, and S. A. Chien. Using generic preferences to incrementally improve plan quality. In *Proc. of the 5th Int'l Conference on Artificial Intelligence Planning and Scheduling (AIPS)*, 236–245, 2000.

[Reiter, 2001] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, Cambridge, MA, 2001.

[Sohrabi and McIlraith, 2008] S. Sohrabi and S. A. McIlraith. On planning with preferences in HTN. In *Proc. of the 12th Int'l Workshop on Non-Monotonic Reasoning (NMR)*, 241–248, 2008.

A Sketch of the Semantics

The satisfaction of all constraint and preference formulae is defined by a translation of formulae into the Situation Calculus (SC), a logical language for reasoning about action and change [Reiter, 2001] Formulae are satified if their translations are entailed by the SC logical theory representing the HTN planning problem and plan. The translation of our HTN constructs are more complex, so we begin with the original elements of PDDL3.

In the SC, primitive actions a are instantaneous. A situation s is a *history* of primitive actions performed at a distinguished initial situation S_0. The logical function $do(a, s)$ returns the situation that corresponds to performing action a in s. In the SC, the *state* of the world is expressed in terms of functions and relations (fluents) relativized to a particular situation s, e.g., $F(\vec{x}, s)$.

The translation to SC proceeds as follows. Since we are operating over finite domains, all universally quantified PDDL3 formulae are translated into individual grounded instances of the formulae. Simple preferences (resp. constraints) are translated into corresponding SC formulae. Temporally extended preferences (resp. constraints) are translated into SC formulae following the translation of LTL formulae into SC by Gabaldon [2004] and Bienvenu *et al.* [2006].

To define the semantics of our HTN extension, we appeal to a translation of HTN planning into SC entailment of a ConGolog program that is again credited to Gabaldon [2002]. ConGolog is a logic programming language built on top of the SC that supports the expression of complex actions. In short, the translation defines a way to construct a logical theory and formula $\Psi(s)$ such that $\Psi(s)$ is entailed by the logical theory iff the sequence of actions encoded by s is a solution to the original HTN planning problem.

More specifically, the initial HTN state s_0 is encoded as the initial situation, S_0. The HTN domain description maps to a corresponding SC domain description, $\mathcal{D}$, where for every operator o there is a corresponding primitive action a, such that the preconditions and the effects of o are axiomatized in $\mathcal{D}$. Every method and nonprimitive task together with constraints is encoded as a ConGolog procedure. $\mathcal{R}$ is the set of procedures in the ConGolog domain theory.

In addition to this translation, we need to deal with the new elements of PDDL3 that we introduced: **occ**(a), **initiate**(X), and **terminate**(X). To this end, following Gabaldon's translation we add two new primitive actions *start*$(P(\vec{v}))$, *end*$(P(\vec{v}))$, to each procedure P that corrensponds to an HTN task or method. In addition, we add the fluents *executing*$(P(\vec{v}), s)$ and *terminated*(X, s), where $P(\vec{v})$ is a ConGolog procedure and X is either $P(\vec{v})$ or a primitive action a. *executing*$(P(\vec{v}), s)$ states that $P(\vec{v})$ is executing in situation s, *terminated*(X, s) states that X has terminated in s. *executing*(a, s) where a is a primitive action, is defined to be false.

occ(a), **initiate**(X), and **terminate**(X) are translated into the situation calculus by building SC formulae that are evaluated when they appear in a preference formula. Below we define these formulae, using a notation compatible with Gabaldon's translation, in which $\varphi[s', s]$ denotes that the (temporal) expression φ holds over the situation fragment s, that starts in situation s'.

occ(a) tells us the first action executed is a:

$$\mathbf{occ}(a)[s', s] = do(a, s') \sqsubseteq s$$

initiate(X) and **terminate**(X) are interpreted as follows:

$$\mathbf{initiate}(X)[s', s] = \begin{cases} do(X, s') \sqsubseteq s & \text{if X} \in \mathcal{A} \\ do(start(X), s') \sqsubseteq s & \text{if X} \in \mathcal{R} \end{cases}$$

$$\mathbf{terminate}(X)[s', s] = \begin{cases} do(X, s') \sqsubseteq s & \text{if X} \in \mathcal{A} \\ do(end(X), s') \sqsubseteq s & \text{if X} \in \mathcal{R}, \end{cases}$$

where $s' \sqsubseteq s$ denotes that situation s' is a predecessor of situation s, and $\mathcal{A}$ is a set containing all primitive actions.

Space precludes a full exposition of the translation. Details provided here and in Section 3, together with the details in Gabaldon [2002] and Bienvenu *et al.* [2006] provide all the pieces.

Defaults in Action: Non-monotonic Reasoning About States in Action Calculi

Hannes Strass and **Michael Thielscher**
Department of Computer Science
Dresden University of Technology
{hannes.strass, mit}@inf.tu-dresden.de

Abstract

We propose a mechanism for default reasoning in action formalisms that allows to make useful assumptions unless information to the contrary. The mechanism is shown to behave properly when actions are performed, in particular we show that it suffices to apply defaults to the initial state. This allows for very simple reasoning, since the defaults need only be applied once and monotonic entailment can thence be used to solve projection problems. We finally consider two simple, natural generalizations of the approach and show that they admit unintuitive conclusions, thereby pointing out directions for further research.

Introduction

This paper is concerned with the combination of two successful approaches to the logical formalization of commonsense reasoning: logics for actions and non-monotonic logics. The present work is by no means the first to join the two; non-monotonic logics have already been used by the reasoning about actions community in the past. After (McCarthy and Hayes 1969) discovered the fundamental problem of determining the non-effects of actions, the frame problem, it was widely believed that non-monotonic reasoning were necessary to solve it. Then (Hanks and McDermott 1987) gave a (by now famous) example of how straightforward use of non-monotonic logics in reasoning about actions and change can lead to counter-intuitive results. When monotonic solutions to the frame problem were found (Reiter 1991; Thielscher 1999), non-monotonic reasoning again seemed to be obsolete.

In this paper, we argue that utilizing default logic still is of use when reasoning about actions. We will not use it to solve the frame problem, however, the solution to the frame problem we use here is monotonic and similar to the one of (Thielscher 1999), but to make useful default assumptions about states.

The approach we propose uses deterministic actions without conditional effects and a restricted form of default assumptions. The main reasoning task we are interested in is the projection problem, that is, given an initial situation and a sequence of actions, the question whether a certain condition holds in the resulting state. The approach can be used to draw intuitive conclusions that are not possible to draw in a monotonic way. As the main result of this paper, we show that default applications can be restricted to the initial state without losing any inferences, thus giving way to a simple reasoning mechanism.

In the second half of the paper, we consider two straightforward generalizations of our approach and show how they permit counterintuitive conclusions, which justifies the restrictions made earlier. The first generalization allows for more general defaults: they are still supernormal, that is, prerequisite-free and normal, but enable default conclusions to be "carried back in time." This clearly disqualifies for solving projection problems, since we would have to take an infinite number of future time points into account. The second generalization allows for more general effect axioms: they are still deterministic but permit to express conditional effects. This again causes conclusions that rely on time points that are intrinsically irrelevant for the question to be answered. In both cases, we identify reasons for the undesired inferences and propose how further work can be done to overcome those shortcomings.

Background

This section introduces the foundations upon which our work rests. Firstly, a unifying action calculus that we will use to axiomatize action domains. Secondly, a restricted version of one of the most prominent non-monotonic logics, Raymond Reiter's Default Logic (Reiter 1980).

The Unifying Action Calculus

Recently, (Thielscher 2009) proposed a unifying action calculus (UAC) with the objective of bundling research efforts in action formalisms. It does not confine to a particular time structure and can thus be instantiated with situation-based action calculi, like the Situation Calculus (McCarthy 1963) or the Fluent Calculus (Thielscher 1999), as well as with formalisms using a linear time structure, like the Event Calculus (Kowalski and Sergot 1986).

The UAC uses only the sorts FLUENT, ACTION, and TIME along with the predicates $<$: TIME × TIME (denoting an ordering of time points), $Holds$: FLUENT × TIME (stating whether a fluent evaluates to true at a given time point), and $Poss$: ACTION × TIME × TIME (indicating whether an action is applicable for particular starting and ending time points). Uniqueness-of-names is assumed for all (finitely many) functions into sorts FLUENT and ACTION.

The following definition introduces the most important types of formulas of the unifying action calculus: they allow to express properties of states and applicability conditions and effects of actions.

Definition 1. Let $\vec{s}$ be a sequence of variables of sort TIME.

- A *state formula* $\Phi[\vec{s}]$ *in* $\vec{s}$ is a first-order formula with free variables $\vec{s}$ where
  - for each occurrence of $Holds(\varphi, s)$ in $\Phi[\vec{s}]$ we have $s \in \vec{s}$ and
  - predicate $Poss$ does not occur.

Let s, t be variables of sort TIME and A be a function into sort ACTION.

- A *precondition axiom* is of the form
$$Poss(A(\vec{x}), s, t) \equiv \pi_A[s] \tag{1}$$
where $\pi_A[s]$ is a state formula in s with free variables among $s, t, \vec{x}$.
- An *effect axiom* is of the form
$$\begin{aligned} &Poss(A(\vec{x}), s, t) \supset \\ &(\forall f)((\gamma_A^+ \vee (Holds(f, s) \wedge \neg\gamma_A^-)) \equiv Holds(f, t)) \end{aligned} \tag{2}$$
where
$$\gamma_A^+ = \bigvee_{0 \le i \le n_A^+} f = \varphi_i \text{ and } \gamma_A^- = \bigvee_{0 \le i \le n_A^-} f = \psi_i$$
and the φ_i and ψ_i are terms of sort FLUENT with free variables among $\vec{x}$.

Readers may be curious as to why the predicate $Poss$ carries two time arguments instead of just one: $Poss(a, s, t)$ is to be read as "action a is possible starting at time s and ending at time t." The formulas γ_A^+ and γ_A^- enumerate the positive and negative effects of the action, respectively. This definition of effect axioms is a restricted version of the original definition of (Thielscher 2009)—it only allows for deterministic actions with unconditional effects.

A few words on notation and naming conventions: lowercase letters will denote object-level variables, we usually use f for sort FLUENT, a for sort ACTION, and s, t for sort TIME. Capital letters and words will denote object level functions of all sorts. Lower-case Greek letters will serve as meta-level variables for fluent and action terms. Capital Greek letters denote formulas or sets of formulas. As usual, $s \le t$ abbreviates $s < t \vee s = t$. Formulas with occurrences of free variables are assumed universally prenex-quantified.

Next, we formalize the concept of an (action) domain axiomatization with its notion of time and action laws.

Definition 2. A *(UAC) domain axiomatization* consists of a finite set of foundational axioms Ω (that define the underlying time structure and do not contain the predicates $Holds$ and $Poss$), a set Π of precondition axioms (1), and a set Υ of effect axioms (2); the latter two for all functions into sort ACTION.

A domain axiomatization is *progressing*, if

- $\Omega \models (\exists s : \text{TIME})(\forall t : \text{TIME}) s \le t$ and
- $\Omega \cup \Pi \models Poss(a, s, t) \supset s < t$.

A domain axiomatization is *sequential*, if it is progressing and

$$\begin{aligned} &\Omega \cup \Pi \models Poss(a, s, t) \wedge Poss(a', s', t') \supset \\ &\qquad (t < t' \supset t \le s') \wedge (t = t' \supset (a = a' \wedge s = s')) \end{aligned}$$

That is, a domain axiomatization is progressing if there exists a least time point and time always increases when applying an action. A sequential domain axiomatization furthermore requires that no two actions overlap.

Lastly, we formalize the intuition of a time point that is reachable via a finite sequence of actions.

Definition 3. Let Σ be a domain axiomatization. A time point τ is *finitely reachable in* Σ iff $\Sigma \models Reach(\tau)$, where the predicate $Reach : \text{TIME}$ is macro-defined by

$$\begin{aligned} Reach(r) &\overset{\text{def}}{=} (\forall R)((\forall s)(Init(s) \supset R(s)) \\ &\quad \wedge (\forall a, s, t)(R(s) \wedge Poss(a, s, t) \supset R(t)) \supset R(r)) \\ Init(t) &\overset{\text{def}}{=} \neg(\exists s) s < t \end{aligned}$$

Note that these macros allow us to perform induction on reachable time points as follows: to show that a certain property $\Psi[s]$ holds for all reachable time points, we show that all minimal time points satisfy the property and that it is preserved by action application to reachable time points.

The examples of this paper will employ situations as their underlying time structure, so we briefly recall the corresponding foundational axioms from (Pirri and Reiter 1999):

$$\neg(s < S_0) \tag{3}$$
$$s < Do(a, s') \equiv s \le s' \tag{4}$$
$$Do(a, s) = Do(a', s') \equiv (a = a' \wedge s = s') \tag{5}$$
$$(\forall P)((P(S_0) \wedge (P(s) \supset P(Do(a, s)))) \supset P(s')) \tag{6}$$

The above axioms shall henceforth be referred to as Ω_{sit}. Whenever we use them as underlying time structure, we stipulate that for each action function A with right hand side $\pi_A[s]$ of precondition axiom (1), we have $\pi_A[s] \equiv \pi'_A[s] \wedge t = Do(A(\vec{x}), s)$ for some π'_A.

Since we are mainly interested in the projection problem, our domain axiomatizations will usually include a set Σ_0 of state formulas in S_0 that characterize the initial situation.

To illustrate the intended usage of the introduced notions, we make use of a variant of a well-known example already mentioned earlier: the Yale Shooting scenario (Hanks and McDermott 1987).

Example 1. Consider the domain axiomatization $\Sigma = \Omega_{sit} \cup \Pi \cup \Upsilon \cup \Sigma_0$. The precondition axioms say that the action Shoot is possible if the gun is loaded and the actions Load and Wait are always possible.

$$\begin{aligned} \Pi = \{&Poss(\mathsf{Shoot}, s, t) \equiv \\ &\qquad (Holds(\mathsf{Loaded}, s) \wedge t = Do(\mathsf{Shoot}, s)), \\ &Poss(\mathsf{Load}, s, t) \equiv t = Do(\mathsf{Load}, s), \\ &Poss(\mathsf{Wait}, s, t) \equiv t = Do(\mathsf{Wait}, s)\} \end{aligned}$$

With these preconditions and foundational axioms (3)–(6), the domain axiomatization is sequential. The effect of shooting is that the turkey ceases to be alive, loading the gun causes it to be loaded, and waiting does not have any effect. All effect axioms in Υ are of the form (2), we state only the $\gamma^{\pm}$ different from the empty disjunction.

$$\gamma^{-}_{\mathsf{Shoot}} = (f = \mathsf{Alive})$$
$$\gamma^{+}_{\mathsf{Load}} = (f = \mathsf{Loaded})$$

Finally, we state that the turkey is alive in the initial situation S_0.

$$\Sigma_0 = \{Holds(\mathsf{Alive}, S_0)\}$$

We can now employ logical entailment to answer the question whether the turkey is still alive after applying the actions Load, Wait, and Shoot, respectively. With the notation $Do([a_1, \dots, a_n], s)$ as abbreviation for $Do(a_n, Do(\dots, Do(a_1, s) \dots))$, it is easy to see that

$$\Sigma \models \neg Holds(\mathsf{Alive}, Do([\mathsf{Load}, \mathsf{Wait}, \mathsf{Shoot}], S_0).$$

Default Logic

Introduced in the seminal paper (Reiter 1980), Default Logic has become one of the most important formalisms for nonmonotonic reasoning. The semantics for supernormal defaults used here is taken from (Brewka and Eiter 1999).Here, a default rule always comes without a prerequisite, and justification and consequence always coincide. A default rule can thus also be seen as a hypothesis that we are willing to assume, but prepared to give up in case of contradiction. A default theory then adds a set of formulas, the *indefeasible knowledge*, that we are not willing to give up for any reason.

Definition 4. A *supernormal default rule*, or, for short, *default*, is a closed first-order formula. Any formulas with occurrences of free variables are taken as representatives of their ground instances.

For a set of closed formulas S, we say the default δ is *active in S* if both $\delta \notin S$ and $\neg\delta \notin S$.

A *(supernormal) default theory* is a pair $(W, \mathcal{D})$, where W is a set of sentences and $\mathcal{D}$ a set of default rules.

An extension for a default theory can be seen as a way of assuming as many defaults as possible without creating inconsistencies. It should be noted that, although the definition differs, our extensions are extensions in Reiter's (1980) sense.

Definition 5. Let $(W, \mathcal{D})$ be a default theory where all default rules are supernormal and $\prec\!\!\!\prec$ be a total order on $\mathcal{D}$. Define $E_0 := Th(W)$ and for all $i > 0$,

$$E_{i+1} = \begin{cases} E_i & \text{if no default is active in } E_i \\ Th(E_i \cup \{\delta\}) & \text{otherwise, where } \delta \text{ is the } \prec\!\!\!\prec\text{-} \\ & \text{minimal default active in } E_i. \end{cases}$$

Then the set $E := \bigcup_{i>0} E_i$ is called the *extension generated by* $\prec\!\!\!\prec$. A set of formulas E is a *preferred extension* for $(W, \mathcal{D})$ if there exists a total order $\prec\!\!\!\prec$ that generates E. The set of all preferred extensions for a default theory $(W, \mathcal{D})$ is denoted by $Ex(W, \mathcal{D})$.

Extensions need not be unique: if there are two contradicting defaults δ and $\neg\delta$, either both or none of them are active in $Th(W)$. Applying one of them makes the other inactive, thus they give rise to two different extensions.

Based on extensions, one can define skeptical and credulous conclusions for default theories: skeptical conclusions are formulas that are contained in every extension, credulous conclusions are those that are contained in at least one extension.

Definition 6. Let $(W, \mathcal{D})$ be a supernormal default theory and Ψ be a first-order formula.

$$W \approx^{skept}_{\mathcal{D}} \Psi \overset{\text{def}}{\equiv} \Psi \in \bigcap_{E \in Ex(W,\mathcal{D})} E$$
$$W \approx^{cred}_{\mathcal{D}} \Psi \overset{\text{def}}{\equiv} \Psi \in \bigcup_{E \in Ex(W,\mathcal{D})} E$$

In the present work, we will primarily be concerned with skeptical reasoning.

Action Domains with Static Defaults

We now combine the hitherto introduced concepts into the notion of a domain axiomatization with defaults. It is essentially a default theory where the set containing the indefeasible knowledge is a domain axiomatization. The defaults are of a restricted form since we allow only static defaults about states.

Definition 7. A *domain axiomatization with defaults* is a pair $(\Sigma, \mathcal{D}[s])$, where Σ is a UAC domain axiomatization and $\mathcal{D}[s]$ is a set of supernormal defaults of the form $Holds(\varphi, s)$ or $\neg Holds(\varphi, s)$ for a fluent φ.

By $\mathcal{D}[\sigma]$ we denote the set of defaults in $\mathcal{D}[s]$ where s has been instantiated by the term σ.

Example 1 (continued)**.** We add a fluent Broken that indicates if the gun does not function properly. Shooting is now only possible if the gun is loaded *and* not broken:

$$\begin{aligned} &Poss(\mathsf{Shoot}, s, t) \equiv \\ &\quad (Holds(\mathsf{Loaded}, s) \wedge \neg Holds(\mathsf{Broken}, s) \\ &\quad \wedge t = Do(\mathsf{Shoot}, s)) \end{aligned}$$

Unless there is information to the contrary, it should be assumed that the gun has no defects. This is expressed by the following default rule:

$$\mathcal{D}[s] = \{\neg Holds(\mathsf{Broken}, s)\}$$

Without the default assumption, it cannot be concluded that the action Shoot is possible after performing Load and Wait since it cannot be inferred that the gun is not broken. Using the abbreviations $S_1 = Do(\mathsf{Load}, S_0)$, $S_2 = Do(\mathsf{Wait}, S_1)$, and $S_3 = Do(\mathsf{Shoot}, S_2)$, we illustrate how the nonmonotonic entailment relation defined earlier enables us to use the default rule to draw the desired conclusion:

$$\Sigma \approx^{skept}_{\mathcal{D}[S_0]} \neg Holds(\mathsf{Broken}, S_2),$$
$$\Sigma \approx^{skept}_{\mathcal{D}[S_0]} Poss(\mathsf{Shoot}, S_2, S_3), \text{ and}$$
$$\Sigma \approx^{skept}_{\mathcal{D}[S_0]} \neg Holds(\mathsf{Alive}, S_3).$$

The default conclusion that the gun works correctly, drawn in S_0, carries over to S_2 and allows to conclude applicability of Shoot in S_2 and its effects on S_3.

In the example just seen, default reasoning could be restricted to the initial situation. As it turns out, this is sufficient for the type of action domain considered here: effect axiom (2) never "removes" information about fluents and thus never makes more defaults active after executing an action. This observation is formalized by the following lemma. It essentially says that to reason about a time point in which an action ends, it makes no difference whether we apply the defaults to the resulting time point or to the time point when the action starts. This holds of course only due to the restricted nature of effect axiom (2).

Lemma 1. *Let $(\Sigma, \mathcal{D}[s])$ be a domain axiomatization with defaults, α be a ground action such that $\Sigma \models Poss(\alpha, \sigma, \tau)$ for some σ, τ : TIME, and let $\Psi[\tau]$ be a state formula in τ. Then*

$$\Sigma \mathrel{|\!\!\approx}^{skept}_{\mathcal{D}[\sigma]} \Psi[\tau] \textit{ iff } \Sigma \mathrel{|\!\!\approx}^{skept}_{\mathcal{D}[\tau]} \Psi[\tau]$$

Proof. (Sketch.) The proof uses structural induction on $\Psi[\tau]$ with $\Psi[\tau] = \mathit{Holds}(\varphi, \tau)$ being the only interesting case. The result is immediate if Σ is inconsistent, so for the following assume that Σ is consistent. If φ is amongst the positive effects of α, then $\Sigma \models \mathit{Holds}(\varphi, \tau)$ and we are done. If φ is no positive effect of α, the conclusion $\mathit{Holds}(\varphi, \tau)$ relies on a default $\mathit{Holds}(\varphi, s) \in \mathcal{D}[s]$ and φ cannot be a negative effect of α (since the conclusion would be impossible otherwise). Since φ is not changed by α, we have that $\mathit{Holds}(\varphi, \sigma) \in E$ iff $\mathit{Holds}(\varphi, \tau) \in E$ for any extension E for $(\Sigma, \mathcal{D}[\sigma])$ or $(\Sigma, \mathcal{D}[\tau])$. □

We next introduce a helpful regression operator which is inspired by the one from (Reiter 1991). It uses the structure of the effect axioms to reduce reasoning about a time point that is the result of applying an action to reasoning about the time point in which the action started.

Definition 8. The operator $\mathcal{R}_\alpha$ maps, for a given action α, a state formula in τ into a state formula in σ as follows.

$$\mathcal{R}_\alpha(\mathit{Holds}(\varphi, \tau)) \stackrel{\text{def}}{=} (\gamma_\alpha^+\{f \mapsto \varphi\} \vee (\mathit{Holds}(\varphi, \sigma) \wedge \neg\gamma_\alpha^-\{f \mapsto \varphi\}))$$

The operator does not change atomic formulas other than Holds statements, and distributes over the first order connectives in the obvious way.

Now whenever an action α is possible and its effect axiom is available, a state formula in the resulting time point and its regression are indeed equivalent.

Proposition 2. *Let α be a ground term of sort* ACTION *and S be a consistent set of closed formulas that contains an effect axiom (2) for action α and where $S \models Poss(\alpha, \sigma, \tau)$ for some σ, τ :* TIME *and let $\Psi[s]$ be a state formula. Then*

$$S \models \Psi[\tau] \equiv \mathcal{R}_\alpha(\Psi)[\sigma]$$

Proof. By structural induction on Ψ. The only interesting case is $\Psi = \mathit{Holds}(\varphi, \tau)$ for some fluent φ. Let I be a model for S.

$I \models \mathit{Holds}(\varphi, \tau)$
iff $I \models (\gamma_\alpha^+\{f \mapsto \varphi\} \vee (\mathit{Holds}(\varphi, \sigma) \wedge \neg\gamma_\alpha^-\{f \mapsto \varphi\}))$
(since $I \models Poss(\alpha, \sigma, \tau)$ and
I is a model for α's effect axiom)
iff $I \models \mathcal{R}_\alpha(\mathit{Holds}(\varphi, \tau))$ (by definition) □

The next theorem says that all *local* conclusions about a finitely reachable time point σ (that is, all conclusions about σ using defaults from $\mathcal{D}[\sigma]$) are exactly the conclusions about σ that we can draw by instantiating the defaults only with the least time point.

Theorem 3. *Let $(\Sigma, \mathcal{D}[s])$ be a progressing domain axiomatization with defaults, λ its least time point, σ :* TIME *be finitely reachable, and $\Psi[\sigma]$ be a state formula. Then*

$$\Sigma \mathrel{|\!\!\approx}^{skept}_{\mathcal{D}[\sigma]} \Psi[\sigma] \textit{ iff } \Sigma \mathrel{|\!\!\approx}^{skept}_{\mathcal{D}[\lambda]} \Psi[\sigma]$$

Proof. By induction on σ. The base case is trivial. For the induction step, assume that $\Sigma \models Poss(\alpha, \sigma, \tau)$.

$\Sigma \mathrel{|\!\!\approx}^{skept}_{\mathcal{D}[\tau]} \Psi[\tau]$
iff $\Sigma \mathrel{|\!\!\approx}^{skept}_{\mathcal{D}[\sigma]} \Psi[\tau]$ (Lemma 1)
iff $\Sigma \mathrel{|\!\!\approx}^{skept}_{\mathcal{D}[\sigma]} \mathcal{R}_\alpha(\Psi)[\sigma]$ (Proposition 2)
iff $\Sigma \mathrel{|\!\!\approx}^{skept}_{\mathcal{D}[\lambda]} \mathcal{R}_\alpha(\Psi)[\sigma]$ (induction hypothesis)
iff $\Sigma \mathrel{|\!\!\approx}^{skept}_{\mathcal{D}[\lambda]} \Psi[\tau]$ (Proposition 2) □

It thus remains to show that local defaults are indeed exhaustive with respect to local conclusions. The next lemma takes a step into this direction: it states that action application does not increase default knowledge about past time points.

Lemma 4. *Let $(\Sigma, \mathcal{D}[s])$ be a domain axiomatization with defaults, α be a ground action such that $\Sigma \models Poss(\alpha, \sigma, \tau)$ for some σ, τ :* TIME*, and let $\Psi[\rho]$ be a state formula in ρ :* TIME *where $\rho \leq \sigma$. Then*

$$\Sigma \mathrel{|\!\!\approx}^{skept}_{\mathcal{D}[\tau]} \Psi[\rho] \textit{ implies } \Sigma \mathrel{|\!\!\approx}^{skept}_{\mathcal{D}[\sigma]} \Psi[\rho]$$

Proof. (Sketch.) We prove the contrapositive. Let $\Sigma \not\mathrel{|\!\!\approx}^{skept}_{\mathcal{D}[\sigma]} \Psi[\rho]$. Then there is an extension E for $(\Sigma, \mathcal{D}[\sigma])$ where $\Psi[\rho] \notin E$. We generate an extension F for $(\Sigma, \mathcal{D}[\tau])$ as follows. Set the ordering $\prec\!\!\prec$ on $\mathcal{D}[\tau]$such that defaults from $\mathcal{D}[\tau] \cap E$ get higher priority than the ones from $\mathcal{D}[\tau] \setminus E$. None of the latter gets applied during generation of F: roughly, if $\delta[\tau] \notin E$ although there is a default $\delta[s] \in \mathcal{D}[s]$, then $\neg\delta[\tau] \in E$. This can be due to either (1) a contradicting action effect or (2) a contradicting default $\neg\delta[s] \in \mathcal{D}[s]$. In case (1), $\neg\delta[\tau] \in Th(\Sigma)$ and $\delta[\tau]$ is inapplicable. For (2), α does not affect $\neg\delta[\sigma]$, thus $\neg\delta[\tau]$ is applicable in $Th(\Sigma)$ and by construction applied in F, which makes $\delta[\tau]$ inapplicable. Now there exists an $E' \subseteq \mathcal{D}[\tau] \cap E$ such that $F = Th(\Sigma \cup E')$, thus any model for E is a model for F. Hence, $\Psi[\rho] \notin F$ and $\Sigma \not\mathrel{|\!\!\approx}^{skept}_{\mathcal{D}[\tau]} \Psi[\rho]$. □

The converse of the lemma does not hold, since an action effect might preclude a default conclusion about the past. The following theorem now says that no sequence of future actions whatsoever can have an impact on conclusions about the present.

Theorem 5. *Let* $(\Sigma, \mathcal{D}[s])$ *be a progressing domain axiomatization with defaults, let* $\Psi[s]$ *be a state formula,* $\sigma \leq \tau$ *be time points, and* σ *be finitely reachable. Then*

$$\Sigma \mathrel{|}\!\approx^{skept}_{\mathcal{D}[\tau]} \Psi[\sigma] \textit{ implies } \Sigma \mathrel{|}\!\approx^{skept}_{\mathcal{D}[\sigma]} \Psi[\sigma]$$

Proof. If τ is not finitely reachable, we have $\Sigma \models \Psi[\sigma]$ and the claim is immediate, so let τ be finitely reachable. We use induction on τ. The base case, $\tau = \sigma$, is obvious. For the induction step, $\Sigma \models Poss(\alpha, \tau, \tau')$ and $\Sigma \mathrel{|}\!\approx^{skept}_{\mathcal{D}[\tau']} \Psi[\sigma]$ imply $\Sigma \mathrel{|}\!\approx^{skept}_{\mathcal{D}[\tau]} \Psi[\sigma]$ by Lemma 4. The induction hypothesis then yields $\Sigma \mathrel{|}\!\approx^{skept}_{\mathcal{D}[\sigma]} \Psi[\sigma]$. □

The final theorem, our main result, now combines Theorems 3 and 5. It states the sufficiency of instantiation with the least time point with respect to conclusions about reachable time points.

Theorem 6. *Let* $(\Sigma, \mathcal{D}[s])$ *be a progressing domain axiomatization with defaults,* λ *be its least time point,* $\Psi[s]$ *be a state formula, and* $\sigma \leq \tau$ *be terms of sort* TIME *where* σ *is finitely reachable. Then*

$$\Sigma \mathrel{|}\!\approx^{skept}_{\mathcal{D}[\tau]} \Psi[\sigma] \textit{ implies } \Sigma \mathrel{|}\!\approx^{skept}_{\mathcal{D}[\lambda]} \Psi[\sigma]$$

Proof. $\Sigma \mathrel{|}\!\approx^{skept}_{\mathcal{D}[\tau]} \Psi[\sigma]$ implies $\Sigma \mathrel{|}\!\approx^{skept}_{\mathcal{D}[\sigma]} \Psi[\sigma]$ by Theorem 5. By Theorem 3, this is the case iff $\Sigma \mathrel{|}\!\approx^{skept}_{\mathcal{D}[\lambda]} \Psi[\sigma]$. □

Generalizations with Undesired Side Effects

In this section, we show some generalizations of the thus far introduced notion of a domain axiomatization with defaults and show how these generalizations clash with our intuitive notion of relevance. The first subsection generalizes the default hypotheses used, and the second subsection generalizes the effect axioms.

Unrestricted Supernormal Defaults

Concluding atomic propositions about the world is not always enough. Sometimes we wish to express defaults of the form "in general, x are y", for example, "in general, paper airplanes fly[1]." Surely, we could instantiate a default $Holds(\mathsf{Flies}(x), s)$ by all objects x which are known to be paper airplanes. But this is by no means elaboration tolerant (McCarthy 1998) and furthermore does not account for previously unknown paper airplanes. We would much rather have a default rule

$$Holds(\mathsf{PaperAirplane}(x), s) \supset Holds(\mathsf{Flies}(x), s) \qquad (7)$$

which is still supernormal and will let us draw the desired conclusion whenever there is no contradicting information. But, unfortunately, allowing disjunctive defaults has unintuitive side effects:

Example 2. Imagine an action $\mathsf{Fold}(x)$ that transforms a sheet of paper x into a paper airplane:

$$Poss(\mathsf{Fold}(x), s, t) \equiv Holds(\mathsf{SheetOfPaper}(x), s)$$
$$\wedge\, t = Do(\mathsf{Fold}(x), s)$$
$$\gamma^+_{\mathsf{Fold}} = (f = \mathsf{PaperAirplane}(x))$$
$$\gamma^-_{\mathsf{Fold}} = (f = \mathsf{SheetOfPaper}(x))$$

Let the domain axiomatization be $\Sigma = \Omega_{sit} \cup \Pi \cup \Upsilon \cup \Sigma_0$ where Π contains the precondition axiom above, Υ contains effect axiom (2) with γ^+_{Fold} and γ^-_{Fold} stated above, and the initial situation is characterized by $\Sigma_0 = \{Holds(\mathsf{SheetOfPaper}(\mathsf{T}), S_0)\}$. The set of defaults $\mathcal{D}[s]$ contains the single default rule (7). Now after folding T into a paper airplane (using the abbreviation $S_1 = Do(\mathsf{Fold}(\mathsf{T}), S_0)$), we can indeed make the desired conclusion that it flies:

$$\Sigma \mathrel{|}\!\approx^{skept}_{\mathcal{D}[S_1]} Holds(\mathsf{Flies}(\mathsf{T}), S_1)$$

So far, so good. But there is another conclusion that we can draw in S_1 and that refers to the past:

$$\Sigma \mathrel{|}\!\approx^{skept}_{\mathcal{D}[S_1]} Holds(\mathsf{Flies}(\mathsf{T}), S_0)$$

Spelled out, the sheet of paper *already flew before it was folded!* Moreover, this conclusion about the initial situation could not be drawn in the initial situation itself without utilizing a future situation:

$$\Sigma \mathrel{|}\!\not\approx^{skept}_{\mathcal{D}[S_0]} Holds(\mathsf{Flies}(\mathsf{T}), S_0)$$

This line of argument could be read as: "If I folded the sheet of paper into a paper airplane, it would fly. Therefore, it flies." This is counterfactual reasoning gone awry. So what happened?

The problem stems from effect axiom (2) and its incorporated solution to the frame problem: since $\mathsf{Flies}(\mathsf{T})$ holds after $\mathsf{Fold}(\mathsf{T})$ but was not a positive effect of the action, according to the effect axiom it must have held beforehand.

A possible remedy for this flaw is the exclusion of certain fluents from the frame assumption, in particular the exclusion of those fluents that are affected by default conclusions. It remains to be investigated how those fluents are to be identified.

Conditional Effects

Let us get back to defaults that are *Holds* statements or negations thereof, but instead increase the expressiveness of the action domain by allowing *conditional effects*. They are modelled as a case distinction on the right hand side of the effect axiom. For each case, the actual formula expressing the effects is identical to (2).

Definition 9. An *effect axiom with conditional effects* is of the form

$$Poss(A(\vec{x}), s, t) \supset \bigvee_{1 \leq i \leq k} (\Phi_i[s] \wedge \Upsilon_i[s, t]) \qquad (8)$$

where $k \geq 1$, and for each $1 \leq i \leq k$,

$$\Upsilon_i[s, t] = (\forall f)(Holds(f, t) \equiv$$
$$(\gamma^+_i \vee (Holds(f, s) \wedge \neg\gamma^-_i))) \qquad (9)$$

[1] Yes, paper airplanes. Birds are not the only objects that should fly by default.

$$\gamma^+ = \bigvee_{0 \le j \le n_i^+} f = \varphi_{ij} \text{ and } \gamma^- = \bigvee_{0 \le j \le n_i^-} f = \psi_{ij}$$

and the φ_{ij} and ψ_{ij} are terms of sort FLUENT with free variables among $\vec{x}$. The $\Phi_i[s]$ are state formulas in s that define the conditions for case i to apply. They are mutually exclusive and the disjunction of them is a tautology—the actions are thus still deterministic.

Conditional effects allow us to further "inspect" a state and base effects upon state properties. This was not possible with effect axiom (2) where all effects were unconditional and the only possibility to inspect the starting state of an action was by precondition axioms.

Example 3. We slightly modify Example 1: the action Shoot is now always possible but breaks an unloaded gun (that works as expected if loaded and not broken).

$$\begin{aligned}
&\mathit{Poss}(\mathsf{Shoot}, s, t) \equiv t = \mathit{Do}(\mathsf{Shoot}, s)\\
&\mathit{Poss}(\mathsf{Shoot}, s, t) \supset\\
&\quad (\neg \mathit{Holds}(\mathsf{Broken}, s) \wedge \mathit{Holds}(\mathsf{Loaded}, s)) \wedge\\
&\qquad (\forall f)(\mathit{Holds}(f, t) \equiv (\mathit{Holds}(f, s) \wedge f \neq \mathsf{Alive}))\\
&\quad \vee\\
&\quad (\mathit{Holds}(\mathsf{Broken}, s) \vee \neg \mathit{Holds}(\mathsf{Loaded}, s)) \wedge\\
&\qquad (\forall f)(\mathit{Holds}(f, t) \equiv (\mathit{Holds}(f, s) \vee f = \mathsf{Broken}))
\end{aligned}$$

With the gun still being not broken by default and $S_1 = \mathit{Do}(\mathsf{Shoot}, S_0)$, we get the following conclusions: by default, the gun is not broken, even after shooting:

$$\Sigma \mathrel{|\!\!\approx}^{skept}_{\mathcal{D}[S_1]} \neg \mathit{Holds}(\mathsf{Broken}, S_1)$$

But then, it must have been loaded in the initial situation (otherwise it would be broken, which it is not):

$$\Sigma \mathrel{|\!\!\approx}^{skept}_{\mathcal{D}[S_1]} \mathit{Holds}(\mathsf{Loaded}, S_0),$$

although this was not known without utilizing a default about a situation in the future:

$$\Sigma \not\mathrel{|\!\!\approx}^{skept}_{\mathcal{D}[S_0]} \mathit{Holds}(\mathsf{Loaded}, S_0).$$

The flaw with this inference is that it makes default conclusions about a fluent whose value is affected by an action at the same time. This somewhat contradicts our intended usage of defaults about states: we originally wanted to express reasonable assumptions about fluents whose values are initially unknown. It does however not seem very reasonable to assume, while not knowing whether the gun is loaded or not, that the gun is still working after firing it when at the same time we know that firing an unloaded gun will break it. Further research will therefore be conducted to incorporate a notion of causality into our framework.

Conclusions and Future Work

The paper investigated the combination of two successful approaches to the logical formalization of commonsense reasoning, logics for actions and non-monotonic logics, and introduced a framework for default reasoning in action formalisms. This is, to the best of our knowledge, the first work to apply defaults to states in theories of actions and change. Thanks to a well-defined form of the employed effect axioms and defaults, the proposed mechanism behaves in an intuitive way. As our main results show, it is even enough to apply default assumptions only to a single time point, namely the least time point, without losing any of the conclusions. On the other hand, naïve generalizations of our approach have been shown to bear the risk of undesired behavior; it is work in progress to extend the framework in a manner that does not allow for counter-intuitive conclusions.

In the future, we aim at integrating a suitably expressive generalization of our approach into the concept of *agent logic programs* (ALPs)—definite logic programs with two special predicates that are evaluated with respect to an underlying domain axiomatization. We intend to augment ALPs by a negation-as-failure operator and combine the answer set semantics for general logic programs (Gelfond and Lifschitz 1991) with a background theory of action to provide a semantics for the augmented language.

References

Brewka, G., and Eiter, T. 1999. Prioritizing Default Logic: Abridged Report. In *Festschrift on the occasion of Prof. Dr. W. Bibel's 60th birthday*. Kluwer.

Gelfond, M., and Lifschitz, V. 1991. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing* 9:365–385.

Hanks, S., and McDermott, D. 1987. Nonmonotonic Logic and Temporal Projection. *Artificial Intelligence* 33(3):379–412.

Kowalski, R. A., and Sergot, M. J. 1986. A Logic-based Calculus of Events. *New Generation Computing* 4(1):67–95.

McCarthy, J., and Hayes, P. J. 1969. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In *Machine Intelligence*, 463–502. Edinburgh University Press.

McCarthy, J. 1963. Situations and Actions and Causal Laws. Stanford Artificial Intelligence Project: Memo 2.

McCarthy, J. 1998. Elaboration Tolerance. In progress.

Pirri, F., and Reiter, R. 1999. Some Contributions to the Metatheory of the Situation Calculus. *Journal of the ACM* 46(3):325–361.

Reiter, R. 1980. A Logic for Default Reasoning. *Artificial Intelligence* 13:81–132.

Reiter, R. 1991. The Frame Problem in the Situation Calculus: A Simple Solution (Sometimes) and a Completeness Result for Goal Regression. In *Artificial Intelligence and Mathematical Theory of Computation – Papers in Honor of John McCarthy*, 359–380. Academic Press.

Thielscher, M. 1999. From Situation Calculus to Fluent Calculus: State Update Axioms as a Solution to the Inferential Frame Problem. *Artificial Intelligence* 111(1–2):277–299.

Thielscher, M. 2009. A Unifying Action Calculus. Artificial Intelligence. To appear.

How Do I Revise My Agent's Action Theory?

Ivan José Varzinczak
Meraka Institute, CSIR
Pretoria, South Africa
ivan.varzinczak@meraka.org.za

Abstract

Logical theories in reasoning about actions may also evolve, and knowledge engineers need revision tools to incorporate new incoming laws about the dynamic environment. We here fill this gap by providing an algorithmic approach for action theory revision. We give a well defined semantics that ensures minimal change, and show correctness of our algorithms w.r.t. the semantic constructions.

Introduction

Like any logical theory, action theories in reasoning about actions may evolve, and thus need revision methods to adequately accommodate new information about the behavior of actions. In (Eiter et al. 2005; Herzig, Perrussel, and Varzinczak 2006; Varzinczak 2008) update and contraction-based methods for action theory repair are defined. Here we continue this important though quite new thread of investigation and develop a minimal change approach for *revising* a domain description.

The motivation is as follows. Consider an agent designed to interact with a coffee machine. Among her beliefs, the agent may know that a coffee is a hot drink, that after buying she gets a coffee, and that with a token it is possible to buy. We can see the agent's beliefs about the behavior of actions in this scenario as a transition system (Figure 1).

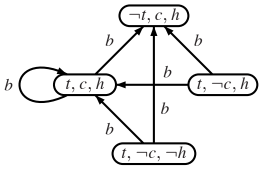

Figure 1: A transition system depicting the agent's knowledge about the dynamics of the coffee machine. b, t, c, and h stand for, respectively, *buy*, *token*, *coffee*, and *hot*.

Well, at some stage the agent may learn that coffee is the only hot drink available at the machine, or that even without a token she can still buy, or that all possible executions of *buy* should lead to states where $\neg$*token* is the case. These are examples of *revision* with new laws about the dynamics of the environment under consideration. And here we are interested in exactly these kinds of theory modification.

The contributions of the present work are as follows:

- What is the semantics of revising an action theory by a law? How to get minimal change, i.e., how to keep as much knowledge about other laws as possible?
- How to syntactically revise an action theory so that its result corresponds to the intended semantics?

Here we answer these questions.

Logical Preliminaries

Our base formalism is multimodal logic K_n (Popkorn 1994).

Action Theories in Multimodal K

Let $\mathfrak{A} = \{a_1, a_2, \ldots\}$ be the set of *atomic actions* of a domain. To each action a there is associated a modal operator $[a]$. $\mathfrak{P} = \{p_1, p_2, \ldots\}$ denotes the set of *propositions*, or *atoms*. $\mathfrak{L} = \{p, \neg p \ : \ p \in \mathfrak{P}\}$ is the set of literals. ℓ denotes a literal and $|\ell|$ the atom in ℓ.

We use $\varphi, \psi, \ldots$ to denote *Boolean formulas*. $\mathfrak{F}$ is the set of all Boolean formulas. A propositional valuation v is a *maximally consistent* set of literals. We denote by $v \Vdash \varphi$ the fact that v satisfies φ. By $val(\varphi)$ we denote the set of all valuations satisfying φ. $\models_{\mathsf{CPL}}$ is the classical consequence relation. $Cn(\varphi)$ denotes all logical consequences of φ.

With $IP(\varphi)$ we denote the set of *prime implicants* (Quine 1952) of φ. By π we denote a prime implicant, and $atm(\pi)$ is the set of atoms occurring in π. Given ℓ and π, $\ell \in \pi$ abbreviates 'ℓ is a literal of π'.

We use $\Phi, \Psi, \ldots$ to denote complex formulas (possibly with modal operators). $\langle a \rangle$ is the dual operator of $[a]$ ($\langle a \rangle \Phi =_{\text{def}} \neg [a] \neg \Phi$).

A K_n-*model* is a tuple $\mathscr{M} = \langle W, R \rangle$ where W is a set of valuations, and R maps action constants a to accessibility relations $R_a \subseteq W \times W$. Given $\mathscr{M}$, $\models^{\mathscr{M}}_{w} p$ (p is true at world w of model $\mathscr{M}$) if $w \Vdash p$; $\models^{\mathscr{M}}_{w} [a]\Phi$ if $\models^{\mathscr{M}}_{w'} \Phi$ for every w' s.t. $(w, w') \in R_a$; truth conditions for the other connectives are as usual. By $\mathcal{M}$ we will denote a set of K_n-models.

$\mathscr{M}$ is a model of Φ (noted $\models^{\mathscr{M}} \Phi$) if and only if $\models^{\mathscr{M}}_{w} \Phi$ for all $w \in W$. $\mathscr{M}$ is a model of a set of formulas Σ (noted $\models^{\mathscr{M}} \Sigma$) if and only if $\models^{\mathscr{M}} \Phi$ for every $\Phi \in \Sigma$. Φ is a *consequence of* *the global axioms* Σ in all K_n-models (noted $\Sigma \models_{\mathsf{K}_n} \Phi$) if and only if for every $\mathscr{M}$, if $\models^{\mathscr{M}} \Sigma$, then $\models^{\mathscr{M}} \Phi$.

In K_n we can state laws describing the behavior of actions. Here we distinguish three types of them.

Static Laws A *static law* is a formula $\varphi \in \mathfrak{F}$ that characterizes the possible states of the world. An example is *coffee* $\rightarrow$ *hot*: if the agent holds a coffee, then she holds a hot drink. The set of static laws of a domain is denoted by $\mathcal{S}$.

Effect Laws An *effect law for a* has the form $\varphi \rightarrow [a]\psi$, with $\varphi,\psi \in \mathfrak{F}$. Effect laws relate an action to its effects, which can be conditional. The consequent ψ is the effect that always obtains when a is executed in a state where the antecedent φ holds. An example is $\mathit{token} \rightarrow [\mathit{buy}]\mathit{hot}$: whenever the agent has a token, after buying, she has a hot drink. If ψ is inconsistent we have a special kind of effect law that we call an *inexecutability law*. For example, $\neg\mathit{token} \rightarrow [\mathit{buy}]\bot$ says that *buy* cannot be executed if the agent has no token. The set of effect laws is denoted by $\mathcal{E}$.

Executability Laws An *executability law for a* has the form $\varphi \rightarrow \langle a \rangle \top$, with $\varphi \in \mathfrak{F}$. It stipulates the context in which a is guaranteed to be executable. (In K_n $\langle a \rangle \top$ reads "a's execution is possible".) For instance, $\mathit{token} \rightarrow \langle \mathit{buy} \rangle \top$ says that buying can be executed whenever the agent has a token. The set of executability laws of a domain is denoted by $\mathcal{X}$.

Given a, $\mathcal{E}_a$ (resp. $\mathcal{X}_a$) will denote the set of only those effect (resp. executability) laws about a.

Action Theories $\mathcal{T} = \mathcal{S} \cup \mathcal{E} \cup \mathcal{X}$ is an *action theory*.

To make the presentation more clear to the reader, we here assume that the agent's theory contains all frame axioms. However, all we shall say here can be defined within a formalism with a solution to the frame and ramification problems like (Herzig, Perrussel, and Varzinczak 2006) do. The action theory of our example will thus be:

$$\mathcal{T} = \left\{ \begin{array}{c} \mathit{coffee} \rightarrow \mathit{hot}, \mathit{token} \rightarrow \langle \mathit{buy} \rangle \top, \\ \neg\mathit{coffee} \rightarrow [\mathit{buy}]\mathit{coffee}, \neg\mathit{token} \rightarrow [\mathit{buy}]\bot, \\ \mathit{coffee} \rightarrow [\mathit{buy}]\mathit{coffee}, \mathit{hot} \rightarrow [\mathit{buy}]\mathit{hot} \end{array} \right\}$$

Figure 1 above shows a K_n-model for the theory $\mathcal{T}$.

Sometimes it will be useful to consider models whose possible worlds are *all* the possible states allowed by $\mathcal{S}$:

Definition 1 *$\mathscr{M} = \langle W, R \rangle$ is a* big frame *of $\mathcal{T}$ if and only if:*

- *$W = \mathit{val}(\mathcal{S})$; and*
- *$R_a = \{(w, w') : \forall.\varphi \rightarrow [a]\psi \in \mathcal{E}_a$, if $\models^{\mathscr{M}}_{w} \varphi$ then $\models^{\mathscr{M}}_{w'} \psi\}$*

Big frames of $\mathcal{T}$ are not always models of $\mathcal{T}$.

Definition 2 *$\mathscr{M}$ is a* supra-model *of $\mathcal{T}$ iff $\models^{\mathscr{M}} \mathcal{T}$ and $\mathscr{M}$ is a big frame of $\mathcal{T}$.*

Figure 2 depicts a supra-model of our example $\mathcal{T}$.

Prime Valuations

An atom p is *essential* to φ if and only if $p \in \mathit{atm}(\varphi')$ for all φ' such that $\models_{\mathsf{CPL}} \varphi \leftrightarrow \varphi'$. For instance, p_1 is essential to $\neg p_1 \wedge (\neg p_1 \vee p_2)$. $\mathit{atm!}(\varphi)$ will denote the essential atoms of φ. (If φ is a tautology or a contradiction, then $\mathit{atm!}(\varphi) = \emptyset$.)

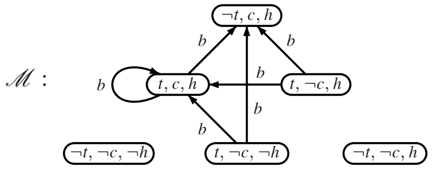

Figure 2: Supra-model for the coffee machine scenario.

For $\varphi \in \mathfrak{F}$, $\varphi*$ is the set of all $\varphi' \in \mathfrak{F}$ such that $\varphi \models_{\mathsf{CPL}} \varphi'$ and $\mathit{atm}(\varphi') \subseteq \mathit{atm!}(\varphi)$. For instance, $p_1 \vee p_2 \notin p_1*$, as $p_1 \models_{\mathsf{CPL}} p_1 \vee p_2$ but $\mathit{atm}(p_1 \vee p_2) \not\subseteq \mathit{atm!}(p_1)$. Clearly, $\mathit{atm}(\bigwedge \varphi*) = \mathit{atm!}(\bigwedge \varphi*)$. Moreover, whenever $\models_{\mathsf{CPL}} \varphi \leftrightarrow \varphi'$, then $\mathit{atm!}(\varphi) = \mathit{atm!}(\varphi')$ and also $\varphi* = \varphi'*$.

Theorem 1 ((Parikh 1999)) *$\models_{\mathsf{CPL}} \varphi \leftrightarrow \bigwedge \varphi*$, and $\mathit{atm}(\varphi*) \subseteq \mathit{atm}(\varphi')$ for every φ' s.t. $\models_{\mathsf{CPL}} \varphi \leftrightarrow \varphi'$.*

Thus for every $\varphi \in \mathfrak{F}$ there is a unique least set of elementary atoms such that φ may equivalently be expressed using only atoms from that set. Hence, $\mathit{Cn}(\varphi) = \mathit{Cn}(\varphi*)$.

Given a valuation v, $v' \subseteq v$ is a *subvaluation*. For W a set of valuations, a subvaluation v' *satisfies* $\varphi \in \mathfrak{F}$ modulo W (noted $v' \Vdash_{W} \varphi$) if and only if $v \Vdash \varphi$ for all $v \in W$ such that $v' \subseteq v$. A subvaluation v *essentially satisfies* φ modulo W ($v \Vdash^{!}_{W} \varphi$) if and only if $v \Vdash_{W} \varphi$ and $\{|\ell| : \ell \in v\} \subseteq \mathit{atm!}(\varphi)$.

Definition 3 *Let $\varphi \in \mathfrak{F}$ and W be a set of valuations. A subvaluation v is a* prime subvaluation *of φ (modulo W) if and only if $v \Vdash^{!}_{W} \varphi$ and there is no $v' \subseteq v$ s.t. $v' \Vdash^{!}_{W} \varphi$.*

A prime subvaluation of a formula φ is one of the weakest states of truth in which φ is true. (Notice the similarity with the syntactical notion of prime implicant (Quine 1952).) We denote all prime subvaluations of φ modulo W by $\mathit{base}(\varphi, W)$.

Theorem 2 *Let $\varphi \in \mathfrak{F}$ and W be a set of valuations. Then for all $w \in W$, $w \Vdash \varphi$ if and only if $w \Vdash \bigvee_{v \in \mathit{base}(\varphi, W)} \bigwedge_{\ell \in v} \ell$.*

Closeness Between Models

When revising a model, we perform a change in its structure. Because there can be several ways of modifying a model (not all minimal), we need a notion of distance between models to identify those closest to the original one.

As we are going to see in more depth in the sequel, changing a model amounts to modifying its possible worlds or its accessibility relation. Hence, the distance between two K_n-models will depend upon the distance between their sets of worlds and accessibility relations. These here will be based on the *symmetric difference* between sets, defined as $X \dot{-} Y = (X \setminus Y) \cup (Y \setminus X)$.

Definition 4 *Let $\mathscr{M} = \langle W, R \rangle$. $\mathscr{M}' = \langle W', R' \rangle$ is* at least as close to *$\mathscr{M}$ as $\mathscr{M}'' = \langle W'', R'' \rangle$, noted $\mathscr{M}' \preceq_{\mathscr{M}} \mathscr{M}''$, iff*

- *either $W \dot{-} W' \subseteq W \dot{-} W''$*
- *or $W \dot{-} W' = W \dot{-} W''$ and $R \dot{-} R' \subseteq R \dot{-} R''$*

This is an extension of Burger and Heidema's relation (Burger and Heidema 2002) to our modal case. Note that other distance notions are also possible, like e.g. the *cardinality* of symmetric differences or Hamming distance.

Semantics of Revision

Contrary to contraction, where we want the negation of a law to be *satisfiable*, in revision we want a new law to be *valid*. Thus we must eliminate all cases satisfying its negation.

The idea in our semantics is as follows: we initially have a set of models $\mathcal{M}$ in which a given formula Φ is (potentially) not valid, i.e., Φ is (possibly) not true in every model in $\mathcal{M}$. In the result we want to have only models of Φ. Adding Φ-models to $\mathcal{M}$ is of no help. Moreover, adding models makes us lose laws: the resulting theory would be more liberal.

One solution amounts to deleting from $\mathcal{M}$ those models that are not Φ-models. Of course removing only some of them does not solve the problem, we must delete every such a model. By doing that, all resulting models will be models of Φ. (This corresponds to *theory expansion*, when the resulting theory is satisfiable.) However, if $\mathcal{M}$ contains no model of Φ, we will end up with $\emptyset$. Consequence: the resulting theory is inconsistent. (This is the main revision problem.) In this case the solution is to *substitute* each model $\mathscr{M}$ in $\mathcal{M}$ by its *nearest modifications* $\mathscr{M}^\star_\Phi$ that makes Φ true. This lets us to keep as close as possible to the original models that we had.

Before defining revision of sets of models, we present what modifications of (individual) models are.

Revising a Model by a Static Law

Suppose that our coffee deliverer agent discovers that the only hot drink that is served on the machine is coffee. In this case, we might want to revise her beliefs with the new static law *coffee* ↔ *hot*.

Considering the model in Figure 2, we see that $\neg$*coffee* $\wedge$ *hot* is satisfiable. As we do not want this, the first step is to *remove* all worlds in which $\neg$*coffee* $\wedge$ *hot* is true. The second step is to guarantee all the remaining worlds satisfy the new law. This issue has been largely addressed in the literature on belief revision and update (Gärdenfors 1988; Winslett 1988; Katsuno and Mendelzon 1992; Herzig and Rifi 1999). Here we can achieve that with a semantics similar to that of classical revision operators: basically one can change the set of possible valuations, by removing or adding worlds.

In our example, removing the possible worlds $\{t, \neg c, h\}$ and $\{\neg t, \neg c, h\}$ would do the job (there is no need to add new valuations since the new static law is satisfied in at least one world of the original model).

The delicate point in removing worlds is that it may result in the loss of some executability laws: in the example, if there were only one arrow leaving some world w and pointing to $\{\neg t, \neg c, h\}$, then removing the latter from the model would make the action under concern no longer executable in w. Here we claim that this is intuitive: if the state of the world to which we could move is no longer possible, then we do not have a transition to that state anymore. Hence, if that transition was the only one we had, it is natural to lose it.

One could also ask what to do with the accessibility relation if new worlds must be added (revision case). We claim that it is reckless to blindly add new elements to R. Instead, we shall postpone correction of executability laws, if needed. This approach is debatable, but with the information we have at hand, it is the safest way of changing static laws.

Definition 5 *Let* $\mathscr{M} = \langle W, R\rangle$. $\mathscr{M}' = \langle W', R'\rangle \in \mathscr{M}^\star_\varphi$ *iff* $W' = (W \setminus val(\neg\varphi)) \cup val(\varphi)$ *and* $R' \subseteq R$.

Clearly $\models^{\mathscr{M}'} \varphi$ for all $\mathscr{M}' \in \mathscr{M}^\star_\varphi$. The minimal models of the revision of $\mathscr{M}$ by φ are those closest to $\mathscr{M}$ w.r.t. $\preceq_{\mathscr{M}}$:

Definition 6 $rev(\mathscr{M}, \varphi) = \bigcup \min\{\mathscr{M}^\star_\varphi, \preceq_{\mathscr{M}}\}$.

In the example of Figure 2, *rev*($\mathscr{M}$, *coffee* ↔ *hot*) is the singleton $\{\mathscr{M}'\}$, with $\mathscr{M}'$ as shown in Figure 3.

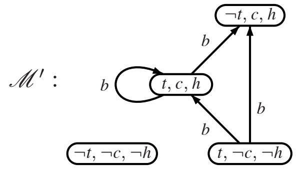

Figure 3: Revising model $\mathscr{M}$ in Figure 2 with *coffee* ↔ *hot*.

Revising a Model by an Effect Law

Let's suppose now that our agent eventually discovers that after buying coffee she does not keep her token. This means that her theory should now be revised by the new effect law *token* → [*buy*]¬*token*. Looking at model $\mathscr{M}$ in Figure 2, this amounts to guaranteeing that *token*∧⟨*buy*⟩*token* is satisfiable in none of its worlds. To do that, we have to look at all the worlds satisfying this formula (if any) and

- either make *token* false in each of these worlds,
- or make ⟨*buy*⟩*token* false in all of them.

If we chose the first option, we will essentially flip the truth value of literal *token* in the respective worlds, which changes the set of valuations of the model. If we chose the latter, we will basically remove *buy*-arrows leading to *token*-worlds, which amounts to changing the accessibility relation.

In our example, worlds $w_1 = \{token, coffee, hot\}$, $w_2 = \{token, \neg coffee, hot\}$ and $w_3 = \{token, \neg coffee, \neg hot\}$ satisfy the formula *token* ∧ ⟨*buy*⟩*token*. Flipping *token* in all of them to ¬*token* would do the job, but this would also have as consequence the introduction of a new static law: ¬*token* would now be valid, i.e., the agent never has a token! Do we want this?

We claim that changing action laws should not have as side effect a change in the static laws. These have a special status (Shanahan 1997), and should change only if required. Hence each world satisfying *token* ∧ ⟨*buy*⟩*token* has to be changed so that ⟨*buy*⟩*token* becomes untrue in it. In the example, we thus should remove (w_1, w_1), (w_2, w_1) and (w_3, w_1) from *R*.

Definition 7 *Let* $\mathscr{M} = \langle W, R\rangle$. $\mathscr{M}' = \langle W', R'\rangle \in \mathscr{M}^\star_{\varphi\to[a]\psi}$ *iff:*

- $W' = W$, $R' \subseteq R$, $\models^{\mathscr{M}'} \varphi \to [a]\psi$, *and*
- *If* $(w, w') \in R \setminus R'$, *then* $\models^{\mathscr{M}}_{w} \varphi$

The minimal models resulting from revision of a model $\mathscr{M}$ by a new effect law are those closest to $\mathscr{M}$ w.r.t. $\preceq_{\mathscr{M}}$:

Definition 8 $rev(\mathscr{M}, \varphi \to [a]\psi) = \bigcup \min\{\mathscr{M}^\star_{\varphi\to[a]\psi}, \preceq_{\mathscr{M}}\}$.

Taking $\mathscr{M}$ as in Figure 2, *rev*($\mathscr{M}$, *token* → [*buy*]¬*token*) will be the singleton $\{\mathscr{M}'\}$ depicted in Figure 4.

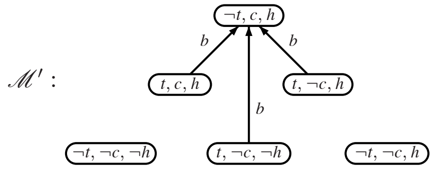

Figure 4: Revising $\mathscr{M}$ in Figure 2 with *token* → [*buy*]¬*token*.

Revising a Model by an Executability Law

Let us now suppose that at some stage it has been decided to grant free coffee to everybody. Faced with this information, we have to revise the agent's laws to reflect the fact that *buy* can also be executed in ¬*token*-contexts: $\neg token \rightarrow \langle buy \rangle \top$ is a new executability law.

Considering model $\mathscr{M}$ in Figure 2, we observe that $\neg token \wedge [buy]\bot$ is satisfiable. Hence we must throw $\neg token \wedge [buy]\bot$ away to ensure the new law becomes true.

To remove $\neg token \wedge [buy]\bot$ we have to look at all worlds satisfying it and modify $\mathscr{M}$ so that they no longer satisfy that formula. Given worlds $w_4 = \{\neg token, \neg coffee, \neg hot\}$ and $w_5 = \{\neg token, \neg coffee, hot\}$, we have two options: change the interpretation of *token* in both or add new arrows leaving these worlds. A question that arises is 'what choice is more drastic: change a world or an arrow'? Again, here we claim that changing the world's content (the valuation) is more drastic, as the existence of such a world is foreseen by some static law and is hence assumed to be as it is, unless we have enough information supporting the contrary, in which case we explicitly change the static laws (see above). Thus we shall add a new *buy*-arrow from each of w_4 and w_5.

Having agreed on that, the issue now is: which worlds should the new arrows point to? In order to comply with minimal change, the new arrows shall point to worlds that are relevant targets of each of the ¬*token*-worlds in question.

Definition 9 *Let $\mathscr{M} = \langle W, R \rangle$, $w, w' \in W$, and $\mathcal{M}$ be a set of models s.t. $\mathscr{M} \in \mathcal{M}$. Then w' is a* relevant target world of w *w.r.t.* $\varphi \rightarrow \langle a \rangle \top$ *for $\mathscr{M}$ in $\mathcal{M}$ iff* $\models^{\mathscr{M}}_{w} \varphi$ *and*

- *If there is $\mathscr{M}' = \langle W', R' \rangle \in \mathcal{M}$ such that $R'_a(w) \neq \emptyset$:*
  - *for all $\ell \in w' \setminus w$, there is $\psi' \in \mathfrak{F}$ s.t. there is $v' \in base(\psi', W)$ s.t. $v' \subseteq w'$, $\ell \in v'$, and $\models^{\mathscr{M}_i}_{w} [a]\psi'$ for every $\mathscr{M}_i \in \mathcal{M}$*
  - *for all $\ell \in w \cap w'$, either there is $\psi' \in \mathfrak{F}$ s.t. there is $v' \in base(\psi', W)$ s.t. $v' \subseteq w'$, $\ell \in v'$, and $\models^{\mathscr{M}_i}_{w} [a]\psi'$ for all $\mathscr{M}_i \in \mathcal{M}$; or there is $\mathscr{M}_i \in \mathcal{M}$ s.t. $\not\models^{\mathscr{M}_i}_{w} [a]\neg\ell$*
- *If $R'_a(w) = \emptyset$ for every $\mathscr{M}' = \langle W', R' \rangle \in \mathcal{M}$:*
  - *for all $\ell \in w' \setminus w$, there is $\mathscr{M}_i = \langle W_i, R_i \rangle \in \mathcal{M}$ s.t. there is $u, v \in W_i$ s.t. $(u, v) \in R_{ia}$ and $\ell \in v \setminus u$*
  - *for all $\ell \in w \cap w'$, there is $\mathscr{M}_i = \langle W_i, R_i \rangle \in \mathcal{M}$ s.t. there is $u, v \in W_i$ s.t. $(u, v) \in R_{ia}$ and $\ell \in u \cap v$, or for all $\mathscr{M}_i = \langle W_i, R_i \rangle \in \mathcal{M}$, if $(u, v) \in R_{ia}$, then $\neg\ell \notin v \setminus u$*

By $rt(w, \varphi \rightarrow \langle a \rangle \top, \mathscr{M}, \mathcal{M})$ we denote the set of all relevant target worlds of w w.r.t. $\varphi \rightarrow \langle a \rangle \top$ for $\mathscr{M}$ in $\mathcal{M}$.

In our example, $w_6 = \{\neg token, coffee, hot\}$ is the only relevant target world here: the two other ¬*token*-worlds violate the effect *coffee* of *buy*, while the three *token*-worlds would make us violate the frame axiom $\neg token \rightarrow [buy]\neg token$.

Definition 10 *Let $\mathscr{M} = \langle W, R \rangle$. $\mathscr{M}' = \langle W', R' \rangle \in \mathscr{M}^{\star}_{\varphi \rightarrow \langle a \rangle \top}$ iff:*

- *$W' = W$, $R \subseteq R'$, $\models^{\mathscr{M}'} \varphi \rightarrow \langle a \rangle \top$, and*
- *If $(w, w') \in R' \setminus R$, then $w' \in rt(w, \varphi \rightarrow [a]\bot, \mathscr{M}, \mathcal{M})$*

The minimal models resulting from revising a model $\mathscr{M}$ by a new executability law are those closest to $\mathscr{M}$ w.r.t. $\preceq_{\mathscr{M}}$:

Definition 11 $rev(\mathscr{M}, \varphi \rightarrow \langle a \rangle \top) = \bigcup \min\{\mathscr{M}^{\star}_{\varphi \rightarrow \langle a \rangle \top}, \preceq_{\mathscr{M}}\}$.

In our running example, $rev(\mathscr{M}, \neg token \rightarrow \langle buy \rangle \top)$ is the singleton $\{\mathscr{M}'\}$, where $\mathscr{M}'$ is as shown in Figure 5.

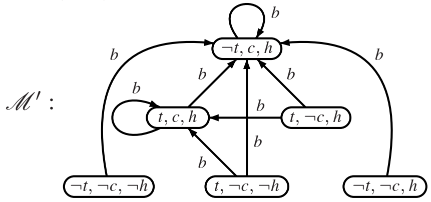

Figure 5: The result of revising model $\mathscr{M}$ in Figure 2 by the new executability law $\neg token \rightarrow \langle buy \rangle \top$.

Revising Sets of Models

Up until now we have seen what the revision of single models means. Now we are ready for a unified definition of revision of a set of models $\mathcal{M}$ by a new law Φ:

Definition 12 *Let $\mathcal{M}$ be a set of models and Φ a law. Then*

$$\mathcal{M}^{\star}_{\Phi} = (\mathcal{M} \setminus \{\mathscr{M} : \not\models^{\mathscr{M}} \Phi\}) \cup \bigcup_{\mathscr{M} \in \mathcal{M}} rev(\mathscr{M}, \Phi)$$

Definition 12 comprises both *expansion* and *revision*: in the former, addition of the new law gives a satisfiable theory; in the latter a deeper change is required to get rid of the inconsistency.

Syntactic Operators for Revision

We now turn our attention to the syntactical counterpart of revision. Our endeavor here is to perform minimal change also at the syntactical level. By $\mathcal{T}^{\star}_{\Phi}$ we denote the result of revising an action theory $\mathcal{T}$ with a new law Φ.

Revising a Theory by a Static Law

Looking at the semantics of revision by Boolean formulas, we see that revising an action theory by a new static law may conflict with the executability laws: some of them may be lost and thus have to be changed as well. The approach here is to preserve as many executability laws as we can in the old possible states. To do that, we look at each possible valuation that is common to the new $\mathcal{S}$ and the old one. Every time an executability used to hold in that state and no inexecutability holds there now, we make the action executable in such a context. For those contexts not allowed by the old $\mathcal{S}$, we make *a* inexecutable (cf. the semantics). Algorithm 1 deals with that ($\mathcal{S} \star \varphi$ denotes the classical revision of $\mathcal{S}$ by φ built upon some well established method from the literature (Winslett 1988; Katsuno and Mendelzon 1992; Herzig and Rifi 1999)).

Algorithm 1 Revision by a Static Law

```
input: $\mathcal{T}, \varphi$
output: $\mathcal{T}^\star_\varphi$
  $\mathcal{S}' := \mathcal{S} \star \varphi, \mathcal{E}' := \mathcal{E}, \mathcal{X}' := \emptyset$
  for all $\pi \in IP(\mathcal{S}')$ do
    for all $A \subseteq \overline{atm(\pi)}$ do
      $\varphi_A := \bigwedge_{\substack{p_i \in \overline{atm(\pi)} \\ p_i \in A}} p_i \wedge \bigwedge_{\substack{p_i \in \overline{atm(\pi)} \\ p_i \notin A}} \neg p_i$
      if $\mathcal{S}' \not\models_{\mathsf{CPL}} (\pi \wedge \varphi_A) \to \bot$ then
        if $\mathcal{S} \not\models_{\mathsf{CPL}} (\pi \wedge \varphi_A) \to \bot$ then
          if $\mathcal{T} \models_{\mathsf{K}_n} (\pi \wedge \varphi_A) \to \langle a \rangle \top$ and $\mathcal{S}', \mathcal{E}', \mathcal{X} \not\models_{\mathsf{K}_n} \neg(\pi \wedge \varphi_A)$ then
            $\mathcal{X}_a{}' := \{(\varphi_i \wedge \pi \wedge \varphi_A) \to \langle a \rangle \top : \varphi_i \to \langle a \rangle \top \in \mathcal{X}_a\}$
        else
          $\mathcal{E}' := \mathcal{E}' \cup \{(\pi \wedge \varphi_A) \to [a]\bot\}$
  $\mathcal{T}^\star_\varphi := \mathcal{S}' \cup \mathcal{E}' \cup \mathcal{X}'$
```

Revising a Theory by an Effect Law

When revising a theory by a new effect law $\varphi \to [a]\psi$, we want to eliminate all possible executions of *a* leading to $\neg\psi$-states. To achieve that, we look at all φ-contexts and every time a transition to some $\neg\psi$-context is not always the case, i.e., $\mathcal{T} \not\models_{\mathsf{K}_n} \varphi \to \langle a \rangle \neg\psi$, we can safely force $[a]\psi$ for that context. On the other hand, if in such a context there is always an execution of *a* to $\neg\psi$, then we should strengthen the executability laws to make room for the new effect in that context we want to add. Algorithm 2 below does the job.

Algorithm 2 Revision by an Effect Law

```
input: $\mathcal{T}, \varphi \to [a]\psi$
output: $\mathcal{T}^\star_{\varphi \to [a]\psi}$
  $\mathcal{T}' := \mathcal{T}$
  for all $\pi \in IP(\mathcal{S} \wedge \varphi)$ do
    for all $A \subseteq \overline{atm(\pi)}$ do
      $\varphi_A := \bigwedge_{\substack{p_i \in \overline{atm(\pi)} \\ p_i \in A}} p_i \wedge \bigwedge_{\substack{p_i \in \overline{atm(\pi)} \\ p_i \notin A}} \neg p_i$
      if $\mathcal{S} \not\models_{\mathsf{CPL}} (\pi \wedge \varphi_A) \to \bot$ then
        for all $\pi' \in IP(\mathcal{S} \wedge \neg \psi)$ do
          if $\mathcal{T}' \models_{\mathsf{K}_n} (\pi \wedge \varphi_A) \to \langle a \rangle \pi'$ then
            $\mathcal{T}' := (\mathcal{T}' \setminus \mathcal{X}'_a) \cup \{(\varphi_i \wedge \neg(\pi \wedge \varphi_A)) \to \langle a \rangle \top : \varphi_i \to \langle a \rangle \top \in \mathcal{X}'_a\}$
        $\mathcal{T}' := \mathcal{T}' \cup \{(\pi \wedge \varphi_A) \to [a]\psi\}$
        if $\mathcal{T}' \not\models_{\mathsf{K}_n} (\pi \wedge \varphi_A) \to [a]\bot$ then
          $\mathcal{T}' := \mathcal{T}' \cup \{(\varphi_i \wedge \pi \wedge \varphi_A) \to \langle a \rangle \top : \varphi_i \to \langle a \rangle \top \in \mathcal{T}\}$
  $\mathcal{T}^\star_{\varphi \to [a]\psi} := \mathcal{T}'$
```

Revising a Theory by an Executability Law

Revision of a theory by a new executability law has as consequence a change in the effect laws: all those laws preventing the execution of *a* shall be weakened. Moreover, to comply with minimal change, we must ensure that in all models of the resulting theory there will be at most *one* transition by *a* from those worlds in which $\mathcal{T}$ precluded *a*'s execution.

Let $(\mathcal{E}_a^{\varphi,\bot})_1, \ldots, (\mathcal{E}_a^{\varphi,\bot})_n$ denote minimum subsets (w.r.t. set inclusion) of $\mathcal{E}_a$ such that $\mathcal{S}, (\mathcal{E}_a^{\varphi,\bot})_i \models_{\mathsf{K}_n} \varphi \to [a]\bot$. (According to (Herzig and Varzinczak 2007), one can ensure at least one such a set always exists.) Let $\mathcal{E}_a^- = \bigcup_{1 \leq i \leq n} (\mathcal{E}_a^{\varphi,\bot})_i$. The effect laws in $\mathcal{E}_a^-$ will serve as guidelines to get rid of $[a]\bot$ in each φ-world allowed by $\mathcal{T}$: they are the laws to be weakened to allow for $\langle a \rangle \top$ in φ-contexts.

Our algorithm works as follows. To force $\varphi \to \langle a \rangle \top$ to be true in all models of the resulting theory, we visit every possible φ-context allowed by it and make the following operations to ensure $\langle a \rangle \top$ is the case for that context: Given a φ-context, if $\mathcal{T}$ does not always preclude *a* from being executed in it, we can safely force $\langle a \rangle \top$ without modifying other laws. On the other hand, if *a* is always inexecutable in that context, then we should weaken the laws in $\mathcal{E}_a^-$. The first thing we must do is to preserve all old effects in all other φ-worlds. To achieve that we specialize the above laws to each possible valuation (maximal conjunction of literals) satisfying φ but the actual one. Then, in the current φ-valuation, we must ensure that action *a* may have any effect, i.e., from this φ-world we can reach any other possible world. We achieve that by weakening the *consequent* of the laws in $\mathcal{E}_a^-$ to the exclusive disjunction of all possible contexts in $\mathcal{T}$. Finally, to get minimal change, we must ensure that all literals in this φ-valuation that are not forced to change are preserved. We do this by stating a conditional frame axiom of the form $(\varphi_k \wedge \ell) \to [a]\ell$, where φ_k is the above-mentioned φ-valuation.

Algorithm 3 gives the pseudo-code for that.

Correctness of the Algorithms

Suppose we have two atoms p_1 and p_2, and one action *a*. Let $\mathcal{T}_1 = \{\neg p_2, p_1 \to [a]p_2, \langle a \rangle \top\}$. The only model of $\mathcal{T}_1$ is $\mathscr{M}$ in Figure 6. Revising such a model by $p_1 \vee p_2$ gives us the models $\mathscr{M}'_i$, $1 \leq i \leq 3$, in Figure 6. Now, revising $\mathcal{T}_1$ by $p_1 \vee p_2$ will give us ${\mathcal{T}_1}^\star_{p_1 \vee p_2} = \{p_1 \wedge \neg p_2, p_1 \to [a]p_2\}$. The only model of ${\mathcal{T}_1}^\star_{p_1 \vee p_2}$ is $\mathscr{M}'_1$ in Figure 6. This means that the semantic revision may produce models (viz. $\mathscr{M}'_2$ and $\mathscr{M}'_3$ in Figure 6) that are not models of the revised theories.

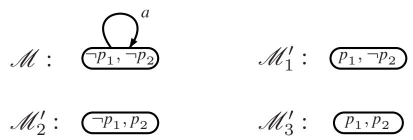

Figure 6: Model $\mathscr{M}$ of $\mathcal{T}_1$ and revision of $\mathscr{M}$ by $p_1 \vee p_2$.

The other way round the algorithms may give theories whose models do not result from revision of models of the initial theory: let $\mathcal{T}_2 = \{(p_1 \vee p_2) \to [a]\bot, \langle a \rangle \top\}$. Its only model is $\mathscr{M}$ (Figure 6). Revising $\mathscr{M}$ by $p_1 \vee p_2$ is as above. However ${\mathcal{T}_2}^\star_{p_1 \vee p_2} = \{p_1 \vee p_2, (p_1 \vee p_2) \to [a]\bot\}$ has a model $\mathscr{M}'' = \langle \{\{p_1, p_2\}, \{p_1, \neg p_2\}, \{\neg p_1, p_2\}\}, \emptyset \rangle$ that is not in $\mathscr{M}^\star_{p_1 \vee p_2}$.

Algorithm 3 Revision by an executability law

```
input: $\mathcal{T}, \varphi \rightarrow \langle a \rangle \top$
output: $\mathcal{T}^{\star}_{\varphi \rightarrow \langle a \rangle \top}$
  $\mathcal{T}' := \mathcal{T}$
  for all $\pi \in IP(\mathcal{S} \wedge \varphi)$ do
    for all $A \subseteq \overline{atm(\pi)}$ do
      $\varphi_A := \bigwedge_{\substack{p_i \in \overline{atm(\pi)} \\ p_i \in A}} p_i \wedge \bigwedge_{\substack{p_i \in \overline{atm(\pi)} \\ p_i \notin A}} \neg p_i$
      if $\mathcal{S} \not\models_{\mathsf{CPL}} (\pi \wedge \varphi_A) \rightarrow \bot$ then
        if $\mathcal{T}' \models_{\mathsf{K}_n} (\pi \wedge \varphi_A) \rightarrow [a]\bot$ then
          $\mathcal{T}' := (\mathcal{T}' \setminus \mathcal{E}'^{-}_{a}) \cup \{(\varphi_i \wedge \neg(\pi \wedge \varphi_A)) \rightarrow [a]\psi_i : \varphi_i \rightarrow [a]\psi_i \in \mathcal{E}'^{-}_{a}\} \cup \{(\varphi_i \wedge \pi \wedge \varphi_A) \rightarrow [a] \bigoplus_{\substack{\pi' \in IP(\mathcal{S}) \\ A' \subseteq \overline{atm(\pi')}}} (\pi' \wedge \varphi_{A'}) : \varphi_i \rightarrow [a]\psi_i \in \mathcal{E}'^{-}_{a}\}$
          for all $L \subseteq \mathfrak{L}$ do
            if $\mathcal{S} \models_{\mathsf{CPL}} (\pi \wedge \varphi_A) \rightarrow \bigwedge_{\ell \in L} \ell$ then
              for all $\ell \in L$ do
                if $\mathcal{T} \models_{\mathsf{K}_n} \ell \rightarrow [a]\bot$ or $(\mathcal{T} \not\models_{\mathsf{K}_n} \ell \rightarrow [a]\neg\ell$ and $\mathcal{T} \models_{\mathsf{K}_n} \ell \rightarrow [a]\ell)$ then
                  $\mathcal{T}' := \mathcal{T}' \cup \{(\pi \wedge \varphi_A \wedge \ell) \rightarrow [a]\ell\}$
        $\mathcal{T}' := \mathcal{T}' \cup \{(\pi \wedge \varphi_A) \rightarrow \langle a \rangle \top\}$
  $\mathcal{T}^{\star}_{\varphi \rightarrow \langle a \rangle \top} := \mathcal{T}'$
```

All this happens because the possible states are not completely characterized by the static laws. Fortunately, concentrating on supra-models of $\mathcal{T}$, we get the right result.

Theorem 3 *If $\mathcal{M} = \{\mathscr{M} : \mathscr{M}$ is a supra-model of $\mathcal{T}\}$ and there is $\mathscr{M}' \in \mathcal{M}$ s.t. $\models^{\mathscr{M}'} \Phi$, then $\bigcup_{\mathscr{M} \in \mathcal{M}} rev(\mathscr{M}, \Phi) \subseteq \mathcal{M}$.*

Then, revision of models of $\mathcal{T}$ by a law Φ in the semantics produces models of the output of the algorithms $\mathcal{T}^{\star}_{\Phi}$:

Theorem 4 *If $\mathcal{M} = \{\mathscr{M} : \mathscr{M}$ is a supra-model of $\mathcal{T}\} \neq \emptyset$, then for every $\mathscr{M}' \in \mathcal{M}^{\star}_{\Phi}$, $\models^{\mathscr{M}'} \mathcal{T}^{\star}_{\Phi}$.*

Also, models of $\mathcal{T}^{\star}_{\Phi}$ result from revision of models of $\mathcal{T}$ by Φ:

Theorem 5 *If $\mathcal{M} = \{\mathscr{M} : \mathscr{M}$ is a supra-model of $\mathcal{T}\} \neq \emptyset$, then for every $\mathscr{M}'$, if $\models^{\mathscr{M}'} \mathcal{T}^{\star}_{\Phi}$, then $\mathscr{M}' \in \mathcal{M}^{\star}_{\Phi}$.*

Sticking to supra-models of $\mathcal{T}$ is not a big deal. We can use the algorithms in (Herzig and Varzinczak 2007) to ensure $\mathcal{T}$ is characterized by its supra-models and that $\mathcal{M} \neq \emptyset$.

Conclusion and Perspectives

The problem of action theory change has only recently received attention in the literature, both in action languages (Baral and Lobo 1997; Eiter et al. 2005) and in modal logic (Herzig, Perrussel, and Varzinczak 2006; Varzinczak 2008).

Here we have studied what revising action theories by a law means, both in the semantics and at the syntactical (algorithmic) level. We have defined a semantics based on distances between models that also captures minimal change w.r.t. the preservation of effects of actions. With our algorithms and the correctness results we have established the link between the semantics and the syntax for theories with supra-models. (Due to page limits, proofs are omitted here.)

Our next step on the subject is analyze the behavior of our operators w.r.t. AGM-like postulates (Alchourrón, Gärdenfors, and Makinson 1985) for modal theories and the relationship between our revision method and contraction. What is known is that Levi identity (Levi 1977), $\mathcal{T}^{\star}_{\Phi} = \mathcal{T}^{-}_{\neg\Phi} \cup \{\Phi\}$, in general does not hold for action laws Φ. The reason is that up to now there is no contraction operator for $\neg\Phi$ where Φ is an action law. Indeed this is the general contraction problem for action theories: contraction of a theory $\mathcal{T}$ by a general formula (like $\neg\Phi$ above) is still an open problem in the area. The definition of a general method will certainly mostly benefit from the semantic modifications we studied here (addition/removal of arrows and worlds).

Given the relationship between modal logics and description logics, a revision method for DL TBoxes would also benefit from the constructions we defined here.

References

Alchourrón, C.; Gärdenfors, P.; and Makinson, D. 1985. On the logic of theory change: Partial meet contraction and revision functions. *J. of Symbolic Logic* 50:510–530.

Baral, C., and Lobo, J. 1997. Defeasible specifications in action theories. In *Proc. IJCAI*, 1441–1446.

Burger, I., and Heidema, J. 2002. Merging inference and conjecture by information. *Synthese* 131(2):223–258.

Eiter, T.; Erdem, E.; Fink, M.; and Senko, J. 2005. Updating action domain descriptions. In *Proc. IJCAI*, 418–423.

Gärdenfors, P. 1988. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. MIT Press.

Herzig, A., and Rifi, O. 1999. Propositional belief base update and minimal change. *Artificial Intelligence* 115(1):107–138.

Herzig, A., and Varzinczak, I. 2007. Metatheory of actions: beyond consistency. *Artificial Intelligence* 171:951–984.

Herzig, A.; Perrussel, L.; and Varzinczak, I. 2006. Elaborating domain descriptions. In *Proc. ECAI*, 397–401.

Katsuno, H., and Mendelzon, A. 1992. On the difference between updating a knowledge base and revising it. In *Belief revision*. Cambridge. 183–203.

Levi, I. 1977. Subjunctives, dispositions and chances. *Synthese* 34:423–455.

Parikh, R. 1999. Beliefs, belief revision, and splitting languages. In *Logic, Language and Computation*, 266–278.

Popkorn, S. 1994. *First Steps in Modal Logic*. Cambridge University Press.

Quine, W. V. O. 1952. The problem of simplifying truth functions. *American Mathematical Monthly* 59:521–531.

Shanahan, M. 1997. *Solving the frame problem*. Cambridge, MA: MIT Press.

Varzinczak, I. 2008. Action theory contraction and minimal change. In *Proc. KR*, 651–661.

Winslett, M.-A. 1988. Reasoning about action using a possible models approach. In *Proc. AAAI*, 89–93.

Progressing Basic Action Theories with Non-Local Effect Actions

Stavros Vassos
Department of Computer Science
University of Toronto
Toronto, Canada
stavros@cs.toronto.edu

Sebastian Sardina
School of Computer Science
RMIT University
Melbourne, Australia
ssardina@cs.rmit.edu.au

Hector Levesque
Department of Computer Science
University of Toronto
Toronto, Canada
hector@cs.toronto.edu

Abstract

In this paper we propose a practical extension to some recent work on the progression of action theories in the situation calculus. In particular, we argue that the assumption of *local-effect actions* is too restrictive for realistic settings. Based on the notion of *safe-range queries* from database theory and *just-in-time* action histories, we present a new type of action theory, called *range-restricted*, that allows actions to have non-local effects with a restricted range. These theories can represent incomplete information in the initial database in terms of *possible closures* for fluents and can be progressed by directly updating the database in an algorithmic manner. We prove the correctness of our method and argue for the applicability of range-restricted theories in realistic settings.

Introduction

One of the requirements for building agents with a proactive behavior is the ability to reason about action and change. The ability to *predict* how the world will be after performing a sequence of actions is the basis for offline automated planning, scheduling, web-service composition, etc. In the situation calculus (McCarthy & Hayes 1969; Reiter 2001) such reasoning problems are examined in the context of the basic action theories (BATs). These are logical theories that specify the preconditions and effects of actions, and an initial database (DB) that represents the initial state of the world before any action has occurred.

A BAT can be used to solve offline problems as well as to equip a situated agent with the ability to *keep track* of the current state of the world. As a BAT is a static entity, in the sense that the axioms do not change over time, the reasoning about the current state is typically carried over using techniques based on *regression*, that transform the queries about the future into queries about the initial state (Reiter 2001). This is an effective choice for some applications, but a poor one for many settings where an agent may act autonomously for long periods of time. In those cases, it is mandatory that the BAT be (periodically) updated so that the initial DB be replaced by a new one reflecting the changes due to the actions that have already occurred. This is identified as the problem of *progression* for BATs (Lin & Reiter 1997).

In general, a DB in a BAT is an unrestricted first-order logical theory that offers great flexibility and expressiveness. The price to pay is high: for most realistic scenarios it is hard to find practical solutions. As far as progression is concerned, it was shown by Lin and Reiter (1997) that the updated DB requires second-order logic in the general case. For this reason, many restrictions on the BATs have been proposed so that the updated DB is first-order representable. It was recently shown that progression is practical provided actions are limited to have local effects only (Vassos, Gerhard, & Levesque 2008).

The restriction on so-called local-effects actions essentially means that all the properties of the world that may be affected by an action are directly specified by the arguments of the action. For example, an action that may affect two boxes box_1 and box_2 that are located next to the agent needs to explicitly mention them in the arguments of the action, e.g., $break(box_1, box_2)$. In that way, *global effects*, which are considered to be one of the reasons why progression may be second-order, are avoided all-together (e.g., the explosion of a bomb affecting all the objects in the world).

Clearly, the local-effect assumption is too restrictive for many realistic scenarios. For instance, the action of moving a container which causes all objects in it to be moved as well cannot be represented. Similarly, the effect of objects being broken when they are near an object that is exploded cannot be captured with local-effect actions. Such type of *indexical*, though not fully global-effect, information arises naturally in many real domains, e.g., consider the case of a non-player-character in a video game that needs to reason about the effects of moving a container object.

In this paper, we extend local-effect BATs to account for such kind of indexical information. To that end, we present what we call *range-restricted* BATs, that allow effects to be non-local but with a restricted range. For such theories, we describe a method for progression such that the new DB is first-order and finite, and we prove that the method is logically correct. To our knowledge, it is the first result on progression for BATs with an infinite domain, incomplete information, and sensing that goes beyond local-effect.

Formal preliminaries

The situation calculus (McCarthy & Hayes 1969) is a first-order logic language with some limited second-order features, designed for representing and reasoning about dynamically changing worlds. A *situation* represents a world history as a sequence of actions. The constant S_0 is used to denote the initial situation where no action has yet been performed; sequences of actions are built using the function *do*: $do(a, s)$ denotes the situation resulting from performing action a in situation s. Relations whose truth values vary from situation to situation are called *fluents*, and are denoted by predicate symbols taking a situation term as their last argument (e.g., $Holding(x, s)$). A special predicate $Poss(a, s)$ is used to state that action a is executable in situation s; and special function $sr(a, s)$ denotes the (binary) sensing outcome of action a when executed in situation s (Scherl & Levesque 2003).

In this paper, we shall restrict our attention to a language $\mathcal{L}$ with a finite number of *relational* fluent symbols (i.e., no functional fluents) that only take arguments of sort object (apart their last situation argument), an infinite number of constant symbols of sort object, and a finite number of function symbols of sort action that take arguments of sort object. We adopt the following notation with subscripts and superscripts: α and a for terms and variables of sort action; σ and s for terms and variables of sort situation; t and x, y, z, w for terms and variables of sort object. Also, we use A for action function symbols, F, G for fluent symbols, and b, c, d, e, o for constants of sort object.

Often we will focus on sentences that refer to a particular situation. For this purpose, for any σ, we define the set of *uniform formulas in* σ to be all those (first-order or second-order) formulas in $\mathcal{L}$ that do not mention any other situation terms except for σ, do not mention *Poss*, and where σ is not used by any quantifier (Lin & Reiter 1997).

Basic action theories

Within the language, one can formulate action theories that describe how the world changes as the result of the available actions. We focus on a variant of the *basic action theories* (BAT) (Reiter 2001) of the following form:[1]

$\mathcal{D} = \mathcal{D}_{ap} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{una} \cup \mathcal{D}_{sr} \cup \mathcal{D}_0 \cup \mathcal{D}_{fnd} \cup \mathcal{E}$, where:

1. $\mathcal{D}_{ap}$ is the set of action precondition axioms (PAs), one per action symbol A, of the form $Poss(A(\vec{y}), s) \equiv \Pi_A(\vec{y}, s)$, where $\Pi_A(\vec{y}, s)$ is uniform in s.
2. $\mathcal{D}_{ss}$ is the set of successor state axioms (SSAs), one per fluent symbol F, of the form $F(\vec{x}, do(a, s)) \equiv \Phi_F(\vec{x}, a, s)$, where $\Phi_F(\vec{x}, a, s)$ is uniform in s. SSAs capture the effects, and non-effects, of actions.
3. $\mathcal{D}_{sr}$ is the set of sensing-result axioms (SRAs), one for each action symbol A, of the form $sr(A(\vec{y}), s) = r \equiv \Theta_A(\vec{y}, r, s)$, where $\Theta_A(\vec{y}, r, s)$ is uniform in s. SRAs relate sensing outcomes with fluents.
4. $\mathcal{D}_{una}$ is the set of unique-names axioms for actions.
5. $\mathcal{D}_0$, the *initial database (DB)*, is a set of sentences uniform in S_0 that describe the initial situation S_0.
6. $\mathcal{D}_{fnd}$ is the set of domain independent axioms of the situation calculus, formally defining the legal situations.
7. $\mathcal{E}$ is a set of unique-names axioms for object constants.

Progression

We follow the definition of the so-called *strong progression* of (Vassos, Gerhard, & Levesque 2008); we only extend it slightly to account for sensing actions.

Let $\mathcal{D}$ be a BAT over relational fluents $F_1, \ldots, F_n$, and let $Q_1, \ldots, Q_n$ be second-order predicate variables. For any formula ϕ in $\mathcal{L}$, let $\phi\langle\vec{F} : \vec{Q}\rangle$ be the formula that results from replacing any fluent atom $F_i(t_1, \ldots, t_n, \sigma)$ in ϕ, where σ is a situation term, with atom $Q_i(t_1, \ldots, t_n)$.

Definition 1. Let $\mathcal{D}$ be a BAT over fluents $\vec{F}$, α an action of the form $A(\vec{c})$, and d a sensing result. Then, $Pro(\mathcal{D},\alpha,d)$ is the following second-order sentence uniform in $do(\alpha, S_0)$:

$$\exists\vec{Q}.\ \mathcal{D}_0\langle\vec{F} : \vec{Q}\rangle \wedge \Theta_A(\vec{c}, d, do(\alpha, S_0)) \wedge$$
$$\bigwedge_{i=1}^{n} \forall\vec{x}.\ F_i(\vec{x}, do(\alpha, S_0)) \equiv \big(\Phi_i(\vec{x}, \alpha, S_0)\langle\vec{F}:\vec{Q}\rangle\big).$$

We say that a set of formulas $\mathcal{D}_\alpha$ uniform in $do(\alpha, S_0)$ is a *strong progression* of $\mathcal{D}$ wrt (α, d) iff $\mathcal{D}_\alpha$ is logically equivalent to $Pro(\mathcal{D}, \alpha, d)$. ■

The important property of strong progression is that $\mathcal{D}_\alpha \cup (\mathcal{D} - \mathcal{D}_0)$ is equivalent to the original theory $\mathcal{D}$ wrt answering *unrestricted* queries about $do(\alpha, S_0)$ and the future situations after $do(\alpha, S_0)$, even queries that quantify over situations. Although $Pro(\mathcal{D}, \alpha, e)$ is defined in second-order logic we are interested in cases where we can find a $\mathcal{D}_\alpha$ that is *first-order representable*. In the sequel, we shall present a restriction on $\mathcal{D}$ that is a sufficient condition for doing this as well as a method for computing a finite $\mathcal{D}_\alpha$.

Range-restricted basic action theories

In this section we present a new type of basic action theories such that $\mathcal{D}_0$ is a *database of possible closures* and the axioms in $\mathcal{D}_{ap}$, $\mathcal{D}_{ss}$, and $\mathcal{D}_{sr}$ are built on *range-restricted* formulas.

A database of possible closures

Intuitively, we treat each fluent as a *multi-valued function*, where the last argument of sort object is considered as the "*output*" and the rest of the arguments of sort object as the "*input*" of the function.[2] This distinction then is important as we require that $\mathcal{D}_0$ expresses incomplete information only about the output of fluents.

Definition 2. Let $V = \{e_1, \ldots, e_m\}$ be a set of constants and τ a fluent atom of the form $F(\vec{c}, w, S_0)$, where $\vec{c}$ is a vector of constants and w a variable. We say that τ has the *ground input* $\vec{c}$ and the *output* w. The *atomic closure* χ of τ on $\{e_1, \ldots, e_m\}$ is the following sentence:

$$\forall w.F(\vec{c}, w, S_0) \equiv (w = e_1 \vee \cdots \vee w = e_m).$$

The notion generalizes to the vector of atoms $\vec{\tau}$ and the vector of sets of constants $\vec{V}$, as the conjunction of each of the atomic closures of τ_i on V_i. A *possible closures axiom (PCA)* for $\vec{\tau}$ is a disjunction of closures of $\vec{\tau}$. We say that each atomic closure mentioned in the PCA is a *possible closure* wrt the PCA. ■

The following is a straightforward property of closures.

Lemma 1. *Let ϕ be the closure of $\vec{\tau}$ and ψ be a closure of $\vec{\pi}$ on some appropriate vectors. Then $\phi \wedge \psi$ is a consistent closure iff for every i, j such that $\tau_i = \pi_j$, the atomic closure of τ_i in ϕ and the one of π_j in ψ are identical.*

[1] For legibility, we typically omit leading universal quantifiers.

[2] The notion of input-output arguments is similar to that of *modes* in logic programming (Apt & Pellegrini 1994). Also, the results obtained here generalize easily to multiple outputs.

A closure of $\vec{\tau}$ expresses complete information about the output of $\vec{\tau}$ while a PCA for $\vec{\tau}$ expresses disjunctive information it. For example, let $\mathit{Near}(x, y, s)$ represent that y is lying near the object x, and χ_1 be $\forall w.\mathit{Near}(\mathit{bomb}, w, S_0) \equiv (w = \mathit{agent} \vee w = \mathit{box}_1)$. Then, χ_1 is the atomic closure of $\mathit{Near}(\mathit{bomb}, w, S_0)$ on $\{\mathit{agent}, \mathit{box}_1\}$ which states that there are *exactly* two objects near the bomb, namely *agent* and box_1. Similarly, let χ_2 be the closure of $\mathit{Near}(\mathit{bomb}, w, S_0)$ on $\{\mathit{agent}, \mathit{box}_2\}$. Then, $\chi_1 \vee \chi_2$ is a PCA for $\mathit{Near}(\mathit{bomb}, w, S_0)$ expressing that there are exactly two objects near the bomb, one being the agent and the other being either box_1 or box_2.

Next, let us define the form of the initial database $\mathcal{D}_0$.

Definition 3. A *database of possible closures (DBPC)* is a finite set of PCAs such that there is no fluent atom with a ground input that appears in more than one PCA. ■

This implies that for every fluent atom τ with a ground input, either the output of τ is completely unknown in S_0 or there is a finite list of possible closures for τ that are explicitly listed in exactly one PCA.

Going back to the bomb example, let $\mathit{Status}(x, y, s)$ represent that the object x has the status y and let $\mathcal{D}_0$ be the following DBPC: $\{\chi_1 \vee \chi_2, \chi_3, \chi_4, \chi_5\}$, where χ_3 is the closure of $\mathit{Status}(\mathit{agent}, w, S_0)$ on $\{\mathit{ready}\}$, χ_4 the closure of $\mathit{Status}(\mathit{box}_1, w, S_0)$ on $\{\mathit{closed}\}$, χ_5 the closure of $\mathit{Status}(\mathit{box}_2, w, S_0)$ on $\{\mathit{closed}, \mathit{broken}\}$, and χ_1, χ_2 as before. Each sentence in $\mathcal{D}_0$ is a PCA: $\chi_1 \vee \chi_2$ lists two possible closures for $\mathit{Near}(\mathit{bomb}, w, S_0)$, while χ_3, χ_4, χ_5 list one possible closure and express complete information.

We now turn our attention to the so-called *possible answers* to a query $\gamma(\vec{x})$ wrt a DBPC $\mathcal{D}_0$.

Definition 4. Let $\mathcal{D}_0$ be a DBPC, and $\gamma(\vec{x})$ a first-order formula uniform in S_0 whose only free variables are in $\vec{x}$. The *possible answers* to γ wrt $\mathcal{D}_0$, denoted as $\mathsf{pans}(\gamma, \mathcal{D}_0)$, is the smallest set of pairs $(\vec{c}, \chi)$ such that:

- χ is a closure of some vector $\vec{\tau}$ s.t. $\mathcal{E} \cup \{\chi\} \models \gamma(\vec{c})$;
- χ is consistent with $\mathcal{D}_0$ and minimal in the sense that every atomic closure in χ is necessary. ■

Intuitively, $\mathsf{pans}(\gamma, \mathcal{D}_0)$ is a way to characterize all the cases where the query formula $\gamma(\vec{x})$ is satisfied in a model of $\mathcal{D}_0$ for some instantiation of $\vec{x}$. For example, let $\gamma(x)$ be the query $\mathit{Near}(\mathit{bomb}, x, S_0)$. Then, $\mathsf{pans}(\gamma, \mathcal{D}_0)$ is the set $\{(\mathit{agent}, \chi_1), (\mathit{box}_1, \chi_1), (\mathit{agent}, \chi_2), (\mathit{box}_2, \chi_2)\}$.

It is important to observe that the possible answers to a query may be *infinite*. For instance, let $\gamma_1(x)$ be $\mathit{Near}(\mathit{agent}, x, S_0)$. Since nothing is said about the objects near the agent in $\mathcal{D}_0$, for every constant c in $\mathcal{L}$, $(c, \chi_c) \in \mathsf{pans}(\gamma_1(x), \mathcal{D}_0)$, where χ_c is the closure of $\mathit{Near}(\mathit{agent}, w, S_0)$ on $\{c\}$, i.e., there is always a model in which $\mathit{Near}(\mathit{agent}, c)$ would indeed hold. Similarly, let $\gamma_2(x)$ be $\neg \mathit{Near}(\mathit{bomb}, x, S_0)$. Then, $\mathsf{pans}(\gamma_2(x), \mathcal{D}_0)$ includes the infinite set $\{(c, \chi_1) \mid c \neq \mathit{agent}, c \neq \mathit{box}_1\}$, since everything but *agent* or box_1 is far when χ_1 is assumed.

Formulas with finite possible answers

We distinguish two ways that the set of possible answers can be infinite. In the query γ_1 above, this happens because what is being queried is *completely* unknown in $\mathcal{D}_0$. In the second query though, fluent atom $\mathit{Near}(\mathit{bomb}, w)$ is mentioned in some PCA and the infinite number of instantiations c for x are in fact due to the possible closure χ_4 of the PCA.

Our objective is to use $\mathcal{D}_0$ to answer queries for which the possible answers depend on the information that is explicitly expressed in the PCAs. This is captured with the following *just-in-time* assumption for formulas.

Definition 5. Let $\mathcal{D}_0$ be a DBPC and $\gamma(\vec{x})$ a first-order formula uniform in S_0 whose only free variables are in $\vec{x}$. Then $\gamma(\vec{x})$ is *just-in-time (JIT)* wrt $\mathcal{D}_0$ iff for every vector of constants $\vec{c}$, $\gamma(\vec{c})$ is consistent with $\mathcal{D}_0 \cup \mathcal{E}$ iff there exists a closure χ such that $\{\chi\} \cup \mathcal{E} \models \gamma(\vec{c})$, where χ is a conjunction of closures such that each conjunct is a possible closure wrt a PCA in $\mathcal{D}_0$. ■

Assuming that a formula is JIT is not enough to avoid an infinite set of possible answers. We need also to ensure that it is *range-restricted* in the following sense.

Definition 6. The situation-suppressed formula γ in $\mathcal{L}$ is *safe-range* wrt a set of variables X according to the rules:

1. let $\vec{c}, \vec{c}_1, \vec{c}_2$ be a vectors of constants, c, d constants, and x, y distinct variables, then:
   - $x = c$ is safe-range wrt $\{x\}$;
   - $F(\vec{c}, d, S_0), F(\vec{c}_1, x, \vec{c}_2, d, S_0)$ are safe-rage wrt $\{\}$;
   - $F(\vec{c}, y, S_0), F(\vec{c}_1, x, \vec{c}_2, y, S_0)$ are safe-range wrt $\{y\}$;
2. if ϕ is safe-range wrt X_ϕ, ψ is safe-range wrt X_ψ then,
   - $\phi \vee \psi$ is safe-range wrt $X_\phi \cap X_\psi$;
   - $\phi \wedge \psi$ is safe-range wrt $X_\phi \cup X_\psi$;
   - $\neg\phi$ is safe-range wrt $\{\}$;
   - $\exists x \phi$ is safe-range wrt $X/\{x\}$ provided that $x \in X$;
3. no other formula is safe-range.

A formula is said to be *range-restricted* iff it is safe-range wrt the set of its free variables. ■

For example, the formula $\mathit{Near}(x, y, S_0)$ is safe-range wrt $\{y\}$, but not range-restricted and not JIT wrt the $\mathcal{D}_0$ of our example. The formulas $\mathit{Near}(\mathit{bomb}, y, S_0)$ and $\mathit{Near}(\mathit{bomb}, y, S_0) \wedge \mathit{Status}(y, z, S_0)$ are range-restricted as well as JIT wrt $\mathcal{D}_0$.

We now state the main result of this section.

Theorem 1. *Let $\mathcal{D}_0$ be a DBPC and $\gamma(\vec{x})$ a first-order formula uniform in S_0 that is range-restricted and just-in-time wrt $\mathcal{D}_0$. Then, $\mathsf{pans}(\gamma, \mathcal{D}_0)$ is a* finite *set $\{(\vec{c}_1, \chi_1), \ldots, (\vec{c}_n, \chi_n)\}$ such that the following holds:*

$$\mathcal{D}_0 \cup \mathcal{E} \models \forall \vec{x}.\gamma(\vec{x}) \equiv \bigvee_{i=1}^{n} (\vec{x} = \vec{c}_i \wedge \chi_i).$$

Proof sketch. It suffices to prove a stronger lemma about the safe-range formulas as follows. Let $\gamma(\vec{x}, \vec{y})$ be a first-order formula that is just-in-time wrt $\mathcal{D}_0$, safe-range wrt the variables in $\vec{x}$, and does not mention any free variable other than $\vec{x}, \vec{y}$. Then for every constant vector $\vec{d}$ that has the same size as $\vec{y}$, $\mathsf{pans}(\gamma(\vec{x}, \vec{d}), \mathcal{D}_0)$ is a finite set $\{(\vec{e}_1, \chi_1), \ldots, (\vec{e}_n, \chi_n)\}$ such that the following holds:

$$\mathcal{D}_0 \cup \mathcal{E} \models \forall \vec{x}.\gamma(\vec{x}, \vec{d}) \equiv \bigvee_{i=1}^{n} (\vec{x} = \vec{e}_i \wedge \chi_i).$$

We prove this lemma by induction on the construction of the formulas γ. Since γ is safe-range wrt the variables in $\vec{x}$ we only need to consider the cases of the Definition 6. Due to space limitations we only show the case that $\gamma(x, y)$ is $F(\vec{c}_1, y, \vec{c}_2, x)$. Let d be an arbitrary constant of the language. Then $\gamma(x, d)$ is the formula $F(\vec{c}_1, d, \vec{c}_2, x)$. By the fact that $\gamma(x, y)$ is JIT wrt $\mathcal{D}_0$ it is not difficult to show that there is a PCA ϕ in $\mathcal{D}_0$ that mentions $F(\vec{c}_1, d, \vec{c}_2, w)$. Without loss of generality we assume that ϕ is a PCA for $F(\vec{c}_1, d, \vec{c}_2, w)$. We will show how to rewrite ϕ in the form that the lemma requires. The axiom ϕ has the form $\bigvee_{i=1}^{n} \chi_i$, where each χ_i is an atomic closure of $F(\vec{c}_1, d, \vec{c}_2, w)$ on some set of constants $\{e_1, \ldots, e_m\}$, i.e, a sentence of the form $\forall w.F(\vec{c}_1, d, \vec{c}_2, w) \equiv w = e_1 \vee \ldots \vee w = e_m$. For each χ_i of this form let χ'_i be the formula $\bigvee_{j=1}^{m}(x = e_j \wedge \chi_i)$, and let ϕ' be $\forall x.F(\vec{c}_1, d, \vec{c}_2, x) \equiv \bigvee_{i=1}^{n} \chi'_i$. It suffices to show that $\mathcal{D}_0 \cup \mathcal{E} \models \phi'$. Let M be an arbitrary model of $\mathcal{D}_0 \cup \mathcal{E}$. Since ϕ is a sentence in $\mathcal{D}_0$ it follows that $M \models \phi$. By the definition of a possible closures axiom and the Lemma 1 it follows that there is exactly k, $1 \leq k \leq n$, such that $M \models \chi_k$. Observe that if we simplify χ_k to *true* and all the other χ_i to *false* in ϕ' we obtain the sentence χ_k. Therefore, $M \models \phi'$ and since M was an arbitrary model of $\mathcal{D}_0 \cup \mathcal{E}$, it follows that $\mathcal{D}_0 \cup \mathcal{E} \models \phi'$. Also, by the Definition 4 and the structure of ϕ' it follows that the set $\mathsf{pans}(\gamma(x, d), \mathcal{D}_0)$ is the set that the lemma requires. □

In other words, the range-restricted and the JIT assumptions on queries are sufficient conditions to guarantee *finitely* many possible answers. The idea then is to build action theories from range-restricted formulas and allow progression to take place only when the JIT assumption also holds. In this case we shall show in the next session that we are able to effectively progress $\mathcal{D}_0$ in a logically correct way.

First, we assume that the formulas $\Phi_F(\vec{x}, a, s)$ of SSAs have the usual general form (Reiter 2001):

$$\gamma_F^+(\vec{x}, a, s) \vee (F(\vec{x}, s) \wedge \neg\gamma_F^-(\vec{x}, a, s)),$$

where γ_F^+ and γ_F^- characterize the positive and negative effects of actions. A *range-restricted* BAT is built on formulas such that when instantiated with any action argument α and any sensing result e, they become range-restricted.

Definition 7. An SSA for F is *range-restricted* iff $\gamma_F^+(\vec{x}, a, s)$ and $\gamma_F^-(\vec{x}, a, s)$ are disjunctions of formulas of the form:

$$\exists\vec{z}(a = A(\vec{y}) \wedge \phi(\vec{y}, \vec{w}, s)),$$

where $\vec{z}$ corresponds to the variables in $\vec{y}$ but not in $\vec{x}$, $\vec{w}$ to the ones in $\vec{x}$ but not in $\vec{y}$, and $\phi(\vec{x}, \vec{w}, s)$, called a *context formula*, is such that $\phi(\vec{c}, \vec{w}, S_0)$ is range-restricted for any $\vec{c}$. Similarly, an SRA for A is *range-restricted* iff $\Theta_A(\vec{c}, d, S_0)$ is range-restricted for any $\vec{c}$ and d. A *range-restricted basic action theory (RR-BAT)* is a BAT such that all axioms in $\mathcal{D}_{ss}, \mathcal{D}_{sr}$ are range-restricted and $\mathcal{D}_0$ is a DBPC. ■

For example, consider an SSA for $Status(x_1, x_2, s)$. The context formula in γ_{Status}^+ that refers to the action of the bomb exploding may be as follows:

$$a = expl \wedge\ Near(bomb, x_1, s) \wedge x_2 = broken,$$

This has the effect of setting the "broken" status to all objects near the bomb. Note that the action *expl* has no arguments, and that the context formula is range-restricted. It is easy to verify that the formula is JIT wrt $\mathcal{D}_0$ as well. The same holds for a context formula in γ_{Status}^+ that removes any other status for all the affected objects:

$$a = expl \wedge Near(bomb, x_1, s) \wedge Status(x_1, x_2, s) \wedge x_2 \neq broken.$$

Just-in-time progression

The RR-BATs are defined so that the axioms in $\mathcal{D}_{ss}, \mathcal{D}_{sr}$ are built on range-restricted formulas. We now show that under a *just-in-time* assumption there is a finite set of ground fluent atoms that may be affected. The intuition is that in this case we can progress $\mathcal{D}_0$ by appealing to the techniques in (Vassos, Gerhard, & Levesque 2008) that work when the set of fluents that may be affected is fixed by the action.

The progression method for the general case

The next definition captures the condition under which our method for progression is logically correct.

Definition 8. An RR-BAT $\mathcal{D}$ is *just-in-time (JIT)* wrt the ground action α and the sensing result d iff for all fluent symbols F, $\gamma_F^+(\vec{x}, \alpha, S_0)$ and $\gamma_F^-(\vec{x}, \alpha, S_0)$ are JIT wrt $\mathcal{D}_0$, and $\Theta_A(\vec{c}, d, S_0)$ is JIT wrt $\mathcal{D}_0$, where α is $A(\vec{c})$. ■

We introduce the following notation.

Definition 9. Let $\mathcal{D}$ be an RR-BAT that is JIT wrt the ground action α and the sensing result d. The *context set* of (α, d) wrt $\mathcal{D}$, denoted as $\mathcal{J}$, is the set of all the fluent atoms $F(\vec{e}, w, S_0)$ such that one of the following is true:[3]

1. for some b, χ, the pair $(\langle\vec{e}, b\rangle, \chi)$ is a possible answer to $\gamma_F^*(\langle\vec{x}, w\rangle, \alpha, S_0)$ wrt $\mathcal{D}_0$;
2. for some $\vec{o}, b, \chi$, the pair $(\langle\vec{o}, b\rangle, \chi)$ is a possible answer to $\gamma_F^*(\langle\vec{x}, y\rangle, \alpha, S_0)$ wrt $\mathcal{D}_0$ and $F(\vec{e}, w, S_0)$ appears in χ;
3. for some χ, the pair $(\emptyset, \chi)$ is a possible answer to $\Theta_A(\vec{c}, d, S_0)$ wrt $\mathcal{D}_0$, where α is the term $A(\vec{c})$ and $\emptyset$ the empty vector and $F(\vec{e}, w, S_0)$ appears in χ. ■

Intuitively, the context set $\mathcal{J}$ specifies all those atomic closures that need to be updated after the action is performed (case 1) as well as those on which the change is conditioned on (case 2), and the atomic closures for which some condition is sensed to be true (case 3). The important property of $\mathcal{J}$, which follows from Theorem 1, is that it is a *finite* set.

Lemma 2. *Let $\mathcal{D}$ be an RR-BAT that is JIT wrt the ground action α and the sensing result d. Then the context set of (α, d) wrt $\mathcal{D}$ is a finite set.*

We now define the $\mathcal{J}$-models which provide a way of separating $\mathcal{D}_0$ into two parts: one that remains unaffected after the action is performed and one that needs to be updated.

Definition 10. Let $\mathcal{J} = \{\tau_1, \ldots, \tau_n\}$ be the context set of (α, d) wrt a RR-BAT $\mathcal{D}$. A *$\mathcal{J}$-model* χ is a closure of the vector $\langle\tau_1, \ldots, \tau_n\rangle$ such that for every i, $1 \leq i \leq n$, the atomic closure of τ_i in χ is a possible closure wrt some PCA in $\mathcal{D}_0$. ■

Note that there are finitely many $\mathcal{J}$-models. The disjunction ϕ then of all the $\mathcal{J}$-models is a larger PCA that corresponds to the "cross-product" of the PCAs in $\mathcal{D}_0$ that capture the same information about $\vec{\tau}$. Observe that ϕ corresponds to

[3] Whenever the notation γ^* is used, γ^* can be either γ^+ or γ^-.

the part of $\mathcal{D}_0$ that needs updating. The intuition then is that we can progress $\mathcal{D}_0$ by progressing each of the $\mathcal{J}$-models.

Definition 11. Let $\mathcal{D}$ be an RR-BAT that is JIT wrt the ground action α and sensing result d, $\mathcal{J}$ the context set of (α, d), and χ a $\mathcal{J}$-model, where χ is the closure of $\langle F_1(\vec{c}_1, w, S_0), \ldots, F_n(\vec{c}_n, w, S_0)\rangle$ on $\langle V_1, \ldots, V_n\rangle$. The *progression* of χ wrt (α, d) is the closure $\psi_1 \wedge \cdots \wedge \psi_n$, where ψ_i is the closure of $F_i(\vec{c}_i, w, S_0)$ on $(V_i \cup \Gamma_i^+)/\Gamma_i^-$ and Γ_i^* is the following set of constants:

$$\{e \mid (\langle \vec{c}_i, e\rangle, \omega) \in \mathsf{pans}(\gamma^*_{F_i}(\langle \vec{x}, w\rangle, \alpha, S_0)), \omega \wedge \chi \not\models \bot\}.$$

The $\mathcal{J}$-model χ is *filtered* iff for all possible answers $(\vec{o}, \phi)$ to $\Theta_A(\vec{c}, d, S_0)$ wrt $\mathcal{D}_0$, where $\alpha = A(\vec{c})$, $\chi \wedge \phi$ is inconsistent. ■

Each of the $\mathcal{J}$-models χ is updated based on the possible answers of the formulas γ^*_F in $\mathcal{D}_{ss}$. For every possible answer $(\vec{o}, \omega)$ of the instantiated γ^*_F, the atom $F(\vec{o})$ is either removed or added to the closure provided that the condition ω for the change is consistent with the $\mathcal{J}$-model χ in question. Moreover, a $\mathcal{J}$-model may be filtered if it is not consistent with the conditions that are implied by the sensing result d.

We now state the main result of this section that illustrates how the new database is constructed from $\mathcal{D}_0$.

Theorem 2. *Let $\mathcal{D}$ be an RR-BAT that is consistent and JIT wrt the ground action α and the sensing result d, $\mathcal{J}$ the context set of (α, d) wrt $\mathcal{D}$, $\{\chi_1, \ldots, \chi_n\}$ the set of all the $\mathcal{J}$-models that are not filtered, and $\{\phi_1, \ldots, \phi_m\}$ the set of all PCAs in $\mathcal{D}_0$ that do not have any atoms in common with any $\mathcal{J}$-model. Let $\mathcal{D}_\alpha$ be the following set:*

$$\{\bigvee_{i=1}^{n} \psi_i, \phi_1, \ldots, \phi_m\},$$

where ψ_i is the progression of χ_i wrt (α, d). Then, the set $\mathcal{D}_\alpha(S_0/do(\alpha, S_0))$ is a strong progression of $\mathcal{D}$ wrt (α, d), where $\mathcal{D}_\alpha(\sigma/\sigma')$ denotes the result of replacing every occurrence of σ in every sentence in $\mathcal{D}_\alpha$ by σ'.

Observe that the progression of $\mathcal{D}_0$ is again a DBPC.

A practical case

Our method of progression is based on the ability to compute possible answers. The time complexity of the method, as well as the size of $\mathcal{D}_\alpha$, is dominated by the size of the sentence $\bigvee_i \phi_i$ in Theorem 2. Roughly speaking, we do two things that have a high computational cost: first, we compute $\mathsf{pans}(\gamma, \mathcal{D}_0)$ for formulas γ in $\mathcal{D}_{ss}, \mathcal{D}_{sr}$, and second, we combine the answers in a way that is similar to a cross-product.

In order to give some insight on the practicality of our method, we examine the case that the formulas γ that need to be evaluated are similar to the so-called *conjunctive queries* (Abiteboul, Hull, & Vianu 1994), in particular, formulas of the form $\exists\vec{x}(\phi_1 \wedge \cdots \wedge \phi_n)$, where ϕ_i is a possibly non-ground fluent atom with variables that may not be in $\vec{x}$.

Given a conjunctive query γ as input and a DBPC $\mathcal{D}_0$, Algorithm 1 checks whether γ is range-restricted and JIT wrt $\mathcal{D}_0$, and if so, computes the set $\mathsf{pans}(\gamma, \mathcal{D}_0)$. The algorithm works by selecting a fluent atom for which a finite-range assumption can be made (line 4 & 8), simplifying γ wrt this atom, and recursively finding the possible answers for the simplified formula (line 18) until all atoms in γ have been selected (line 1). Instead of working with vectors of terms, the algorithm computes bindings for all variables.

Algorithm 1 $\mathsf{pans}(\gamma, \mathcal{D}_0)$

```
1:  if $\gamma$ is the empty conjunction then
2:      return $\{(\emptyset, \top)\}$          // query reduced to $\top$
3:  end if
4:  $\Delta = \{F(\vec{c}, t, S_0) \in \gamma \mid F(\vec{c}, w, S_0)$ is mentioned in $\mathcal{D}_0\}$
5:  if $\Delta = \emptyset$ then
6:      return failure          // no fluent to continue
7:  else
8:      Pick $F(\vec{c}, t, S_0) \in \Delta$   // arbitrary selection
9:      $X := \emptyset$                        // init answer set
10:     for all $\chi_F = F(\vec{c}, w, S_0) \equiv w = d_1 \vee \ldots \vee d_n \in \mathcal{D}_0$
        do
11:         if $t$ is a variable then
12:             $\Gamma = \{d_1, \ldots, d_n\}$
13:         else
14:             $\Gamma = \{d_1, \ldots, d_n\} \cap \{t\}$
15:         end if
16:         for all constants $e \in \Gamma$ do
17:             $\theta' := \{t/e \mid t$ is variable$\}$
18:             $Y := \mathsf{pans}(\gamma\theta' \setminus \{F(\vec{c}, e, S_0)\}, \mathcal{D}_0)$
19:             if $Y = failure$ then
20:                 return failure      // propagate failure
21:             else
22:                 $W := \{(\theta\theta', \chi \wedge \chi_F) \mid (\theta, \chi) \in Y,\ \mathcal{D}_0 \cup \{\chi \wedge$
                    $\chi_F\} \not\models \bot\}$          // merge results
23:                 $X := X \cup W$     // update current set
24:             end if
25:         end for
26:     end for
27:     $X := \{(\theta|_{\vec{x}}, \chi) \mid (\theta, \chi) \in X, \vec{x}$ are the free variables in $\gamma\}$
28:     return X
29: end if
```

It turns out that the algorithm is a sound and complete way for computing the possible answers of range-restricted and JIT formulas, when these are conjunctive queries.

Theorem 3. *Let $\mathcal{D}_0$ be a DBPC and γ a first-order conjunctive query uniform in S_0. Then, Algorithm 1 always terminates with inputs γ and $\mathcal{D}_0$, and moreover, if γ is range-restricted and JIT wrt $\mathcal{D}_0$, it returns the set $\mathsf{pans}(\gamma, \mathcal{D}_0)$.*

The conjunctive queries are expressive enough to represent basic features of practical domains. For example, the context formula of γ^+_{Status} that we examined earlier, namely $Near(bomb, x_1, s) \wedge x_2 = broken$, is a simple conjunctive query. As another example consider an agent living in a grid-world, typical of many video games. The agent may reason about its next location $Loc(z, do(a, s))$ after doing action a by using an SSA whose positive effect $\gamma^+_{Loc}(z, a, s)$ contains the following disjunct:

$$a = moveFwd \wedge$$
$$\exists x \exists y(Dir(y, s) \wedge Loc(x, s) \wedge Adj(x, y, z, s) \wedge Clear(z, s)).$$

That is, when moving forward, the agent is in location z if z is the adjacent cell to its current location x towards its current direction y (e.g., north, east), and z is not blocked with an obstacle. Clearly, this positive effect relies on multiple indexical information and action *moveFwd* is not local-effect.

Algorithm 1 can easily be extended to handle equalities as well as negated atoms. The first case can be easily addressed via standard unification procedures. For negative literals the idea is to collect also the set Δ^- of *ground* literals of the form $\neg F(\vec{c}, d, S_0)$ such that $F(\vec{c}, w, S_0)$ is mentioned in $\mathcal{D}_0$. When a negative literal is selected, the algorithm works in the same way as for the ground positive literal except that it iterates over the possible closures of $F(\vec{c}, w, S_0)$ for which $F(\vec{c}, d, S_0)$ is *not* true. (Observe that this is similar to the way logic-programming implementation techniques for *negation as failure* (Apt & Pellegrini 1994).)

Finally, a comment about the complexity of Algorithm 1 and progression. Let ℓ be the size of the largest closure in $\mathcal{D}_0$ and k the maximum number of possible closures in a PCA in $\mathcal{D}_0$. Then, Algorithm 1 runs in time $\mathrm{O}(|\gamma|^{k\ell})$: there are $k\ell$ value-closure pairs to be tested for each atom in γ. With respect to progression, this implies that, in the worst case, the size of the new database $\mathcal{D}_\alpha$ may be exponential to the size of $\mathcal{D}_0$. Nevertheless, we expect the size of $\mathcal{D}_\alpha$ to be manageable in practical scenarios like the previous example, where the expressiveness of γ and $\mathcal{D}_0$ is mostly used to answer queries that require indexical reasoning.

Related and future work

The notion of progression for BATs was first introduced by Lin and Reiter (1997). The version we use here is due to Vassos *et al* (2008) which we extended slightly to account for sensing. Lin and Reiter (1997) suggested some strong syntactic restrictions on the BATs that allow for a first-order progression, while Vassos and Levesque (2008) suggested a restriction on the queries. Liu and Levesque (2005) introduced the *local-effect* assumption for actions when they proposed a weaker version of progression that is logically incomplete, but remains practical. Vassos *et al.* (2008) later showed that under this assumption a correct first-order progression can be computed by updating a finite $\mathcal{D}_0$. Our restriction of Definition 7 is similar. The main difference is that we do not require that the arguments $\vec{x}$ of the fluent F are included in the arguments $\vec{y}$ of the action, thus handling cases like the *moveFwd* example. To stay practical though we had to restrict the structure of $\mathcal{D}_0$. Finally, similar to the notion of progression, Shirazi and Amir (2005) proposed *logical filtering* as a way to progress $\mathcal{D}_0$ and proved that their method is correct for answering uniform queries.

The notion of possible closures is a generalization of the *possible values* of Vassos and Levesque (2007). The notions of the safe-range and range-restricted queries come from the database theory where this form of "safe" queries has been extensively studied (Abiteboul, Hull, & Vianu 1994). The notion of just-in-time formulas was introduced for a different setting in (De Giacomo, Levesque, & Sardina 2001) and, in our case, is also related to the *active domain* of a database (Abiteboul, Hull, & Vianu 1994). Outside of the situation calculus, Thielscher (1999) defined a dual representation for BATs based on state update axioms that explicitly define the direct effects of each action, and investigated progression in this setting. Unlike our work where the sentences in $\mathcal{D}_0$ are replaced with an updated version, there, the update relies on expressing the changes using constraints.

Conclusions

In this paper, we proposed a new type of basic action theories, where the initial description is a set of *possible closures* and the effects of actions have a *restricted range*. For these theories, called *range-restricted*, we presented a method that computes a finite first-order progression by directly updating the initial database, and proved its correctness. To the best of our knowledge, it is the first result on the progression of basic action theories with an infinite domain, incomplete information, and sensing that goes beyond the local-effect assumption. We argue that the type of indexical information that our theories can handle arises naturally in real domains, e.g., when an agent needs to reason about the effects of moving a container. We considered also a practical restriction that is typical in logic-programming, and presented an algorithm for the task that our progression method relies on, namely computing possible answers. Our next step is to evaluate the approach by relying on logic-programming frameworks and recent work on inconsistent/incomplete databases (e.g., Fuxman *et al* (2005)).

References

Abiteboul, S.; Hull, R.; and Vianu, V. 1994. *Foundations of Databases : The Logical Level*. Addison Wesley.

Apt, K., and Pellegrini, A. 1994. On the occur-check free Prolog program. *ACM Toplas* 16(3):687–726.

De Giacomo, G.; Levesque, H. J.; and Sardina, S. 2001. Incremental execution of guarded theories. *Computational Logic* 2(4):495–525.

Fuxman, A.; Fazli, E.; and Miller, R. J. 2005. Conquer: efficient management of inconsistent databases. In *Proc. of SIGMOD-05*, 155–166. ACM Press.

Lin, F., and Reiter, R. 1997. How to progress a database. *Artificial Intelligence* 92(1-2):131–167.

Liu, Y., and Levesque, H. J. 2005. Tractable reasoning with incomplete first-order knowledge in dynamic systems with context-dependent actions. In *Proc. of IJCAI*.

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

Reiter, R. 2001. *Knowledge in Action. Logical Foundations for Specifying and Implementing Dyn. Sys.* MIT Press.

Scherl, R., and Levesque, H. J. 2003. Knowledge, action, and the frame problem. *Artificial Intelligence* 144(1–2):1–39.

Shirazi, A., and Amir, E. 2005. First-order logical filtering. In *Proc. of IJCAI-05*, 589–595.

Thielscher, M. 1999. From situation calculus to fluent calculus: State update axioms as a solution to the inferential frame problem. *Artificial Intelligence* 111(1-2):277–299.

Vassos, S., and Levesque, H. 2007. Progression of situation calculus action theories with incomplete information. In *Proc. IJCAI*, 2024–2029.

Vassos, S., and Levesque, H. J. 2008. On the progression of situation calculus basic action theories: Resolving a 10-year-old conjecture. In *Proc. of AAAI*.

Vassos, S.; Gerhard, L.; and Levesque, H. J. 2008. First-order strong progression for local-effect basic action theories. In *Proc. of KR*, 662–272.