ISSN 1369-1961

Department of
Computer Science

Technical Report No. 758

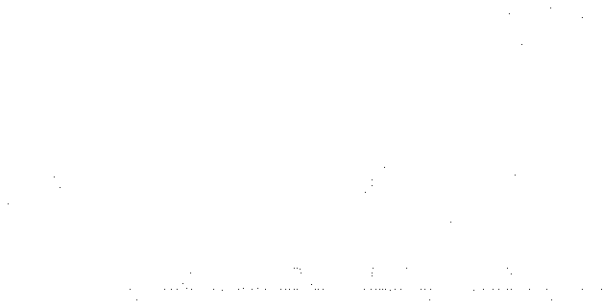
Practical Reasoning and Rationality

Working Notes of the ECAI'98 Workshop

Eds: John Bell &
Zhisheng Huang

QUEEN MARY
AND WESTFIELD COLLEGE
UNIVERSITY OF LONDON

August 1998



Practical Reasoning and Rationality

Working Notes of the ECAI'98 Workshop

Edited by John Bell
and Zhisheng Huang

Working Notes

ECAI'98 Workshop W6

Practical Reasoning and Rationality

24th August 1998
Brighton, England

Organising Committee

John Bell (Queen Mary College, Univ. of London)
Zhisheng Huang (Queen Mary College, Univ. of London)
John-Jules Meyer (Univ. of Utrecht)
Mark Ryan (Univ. of Birmingham)
Marek Sergot (Imperial College, Univ. of London)
Sam Steel (Univ. of Essex)
Mike Wooldridge (Queen Mary College, Univ. of London)

Preface

The theme of this workshop is, intentionally, a broad one. A comprehensive logical theory of practical reasoning and rationality will include theoretical reasoning (reasoning about what is the case, involving informational attitudes such as beliefs and knowledge), practical reasoning (reasoning about what to do, involving motivational attitudes such as desires, goals, intentions, and obligations), and reasoning about actions and their effects.

Recent and current logical work has tended to focus on particular aspects of the problem; including nonmonotonic logics, belief revision, probabilistic logics, argumentation, logics for belief, action, obligation and preference, and logics for reasoning about action and change. This is the third in a series of workshops which aim to promote the integration of this work. Information on the workshop series can be found at: http://www.dcs.qmw.ac.uk/conferences/prr/

We are grateful for the administrative support provided by the ECAI'98 organisers.

Contents

Programme

9.30-11 Session 1

Alessio Lomuscio and Mark Ryan
"Model Refinement and model checking for $S5^n$"

Jan Broersen and Roel Wieringa
"Preferential Semantics for Action Specifications in First-order Modal Action Logic"

Discussion

11.30-1 Session 2

Leendert W.N. van der Torre and Yao-Hua Tan,
"Deliberate Robbery, or the Calculating Samaritan"

Koen V. Hindriks, Frank S. de Boer, Wiebe van der Hoek and John-Jules Meyer, "Formal Semantics of the Core of AGENT-0"

Discussion

2-3.30 Session 3

Munindar P. Singh
"The Dialectic of Practical Reasoning"

Alejandro J. Garcia, Guillermo R. Simari and Carlos I. Chesnevar
"An Argumentative Framework for Reasoning with Inconsistent and Incomplete Information"

Discussion

4-5.30 Session 4

Eliezer L. Lozinskii
"Reasoning by Evidence for Best-Chance Planning with Incomplete Information"

Michael Wooldridge and Simon Parsons
"Intention Reconsideration Reconsidered"

Discussion

Preferential Semantics for Action Specifications in First-order Modal Action Logic[1]

Jan Broersen[2]
Faculty of Mathematics and Computer Science, *Vrije Universiteit*
De Boelelaan 1081a, 1081 HV Amsterdam, the Netherlands
broersen@cs.vu.nl

Roel Wieringa
Faculty of Computer Science, *University of Twente*
P.O. Box 217, 7500 AE Enschede, the Netherlands
roelw@cs.utwente.nl

Abstract

In this paper we investigate preferential semantics for declarative specifications in a First Order Modal Action Logic. We address some well known problems: the frame problem, the qualification problem and the ramification problem. We incorporate the assumptions that are inherent to both the frame and qualification problem into the semantics of the modal Action Logic by defining orderings over Dynamic Logic models. These orderings allow us to identify for each declarative Dynamic Logic action specification a unique intended model.

1 Introduction

Golshani et al. [10] introduced a version of first-order dynamic logic called *Modal Action Logic* (MAL) to specify database updates declaratively. MAL was used later to specify requirements on the external behavior of software and to specify descriptive and prescriptive (deontic) constraints on the external behavior systems in general [14, 13, 12, 20]. Meyer and Wieringa [21, 23, 22] studied a closely related version of *Dynamic Logic* (DL), which was also used to specify descriptive and deontic constraints on database updates and object behavior declaratively. Although these papers show the wide applicability of MAL in the specification of agent behavior, they do not show how we can reason about actions of agents and their effects. In this paper we take a first step towards performing this type of reasoning by defining intended minimal models for a certain class of agent specifications. We argue that these models allow for a form of preferential model checking, which we intend to study in a sequel to this work.

We focus on a specific subset of MAL-formulas to specify the most relevant aspects of (nondeterministic!) actions e.g. their effects, guards on their occurrence and global static invariants. We do not specify sufficient conditions on occurrence of actions. We argue that the role they play can better be handled by a suitable definition of intended (prefered) model in which actions can take place whenever this is compatible with the global invariants and effects. A second problem we address is the problem of minimal change. We define a preference relation over MAL-models that selects the models in which the effects are interpreted under a minimal change condition. We investigate the interplay between both preference relations and define an intended model that is prefered with respect to both orderings. This gives us an exact unique meaning (model) for each specification. The intended model provides the basis for reasoning about properties of the specified actions, by means of model checking.

In section 2, we define our variant of MAL and in section 3, we consider agents subject to a number of *static constraints*, which may be descriptive or prescriptive, and actions that are subject to a number of declarative constraints on their *effects* and declarative *guards* on their occurrence. Corresponding to these two constraints on actions, we define in section 3 two preference relations on models, which order models on minimal change and maximal reachability, respectively. We study properties of these preference relations for three classes of action specifications. In section 4 we compare our results with other work and in section 5, we discuss how our work prepares the way for preferential model checking. This paper extends earlier work done on action specification in *Propositional Dynamic Logic*. In this workshop paper, all proofs are omitted. Most of them are given in the internal report [3] from which this paper is derived.

[1]Partly supported by Esprit Working Group Aspire, contact nr. 22704.
[2]Supported by USF-VU as part of the SINS contract.

2 Modal Action Logic

The syntactic elements of the Modal Action Language we use are the **punctuation symbols:** { ')', '(', ',' }, a set of **variable symbols** $\mathcal{V}$, a set of **predicate symbols** $\mathcal{P}$ including one distinguished predicate denoted by $=$, a set $\mathcal{F}$ of **function symbols**, a set of **propositional connective symbols** $\{\neg, \vee\}$, the **quantification symbol** $\{\exists\}$, a set $\mathcal{A}$ of **atomic action symbols** and the **operation symbol** $\{(.)\}$.

The intended use of the logic is the writing of action specifications for agents. A specific specification uses only a finite subset of the language. A **signature** Σ is thus defined as a specific combination of finite subsets of the predicate, function and atomic action symbols: $\Sigma = (P, F, A)$. Apart from a signature, a specifier will also have to provide arities for predicate and function symbols. The arity of a predicate or function prescribes a length for the term lists concatenated to it. A signature contains the symbols that are to be given an interpretation relative to a specification. All other symbols are given logical interpretations.

The next BNF-sentences (ad hoc extended to incorporate indexing of syntactic elements) define atomic and well formed formulas.

$$
\begin{array}{lcl}
\mathit{term} & ::= & \mathit{variable} \mid \mathit{constant} \mid \mathit{function}^n\ (\ \mathit{termlist}_n\ ) \\
\mathit{constant} & ::= & \mathit{function}^0 \\
\mathit{termlist}_1 & ::= & \mathit{term} \\
\mathit{termlist}_n & ::= & \mathit{termlist}_{n-1}\ ,\ \mathit{term} \\
\mathit{atom} & ::= & \mathit{predicate}^n\ (\ \mathit{termlist}_n\ ) \\
\phi & ::= & \mathit{atom} \mid \top \mid \bot \mid \neg\ \phi \mid \phi\ \vee\ \psi \mid \forall\ \mathit{variable}\ \phi \mid \langle\ \mathit{action}\ \rangle\ \phi
\end{array}
$$

We use $\phi, \psi, \chi, \ldots$ as meta variables over well formed formulas. We apply the usual syntactic abbreviations. A **specification** $Spec = (\Sigma, \Phi)$ is a pair consisting of a signature Σ and a finite set Φ of well formed formulas over Σ.

To define the semantics of MAL we define the notion of MAL-structure.

Definition 1 *Given a signature $\Sigma = (P, F, A)$ a MAL-structure $\mathcal{S} = (S, \mathcal{I}_A, D, \mathcal{I}_P^S, \mathcal{I}_f^S)$ is defined as:*

- *S is a nonempty set of possible states*
- *$\mathcal{I}_A$ is a total function $A \rightarrow 2^{S \times S}$, that maps action symbols to relations over states.*
- *D is an arbitrary domain*
- *$\mathcal{I}_P^S$ is a function $S \rightarrow (P \rightarrow 2^{D^n})$, that for each state s maps predicate symbols to relations over the domain.*
- *$\mathcal{I}_f^S$ is a function $S \rightarrow (F \rightarrow (D^n \rightharpoonup D))$, that for each state s maps function symbols to functions on the domain.*

Combinations of states S and a relation $\mathcal{I}_A$, that are by definition common to many structures, are called 'frames' ($\mathcal{F}$).

We do not want to interpret actions over states that differ in their interpretation of free variables; actions can only influence the interpretation predicate and function symbols. Therefore we require that the assignment of free variables to domain elements is common to all states. This implies that we need a Domain that is common to all states. This is completely different from the situation in Dynamic Logic [11], where actions (programs) are interpreted to modify values (bindings) of free variables. In DL it is the variables that are interpreted different in different states, while the interpretation of predicates, functions and constants stays the same. Note however that function symbols of arity 0 can be used as variables in the sense of DL.

Definition 2 *Given a structure $\mathcal{S} = (S, \mathcal{I}_A, D, \mathcal{I}_P^S, \mathcal{I}_f^S)$ and a set of variables V, an assignment $\mathcal{I}_V$ is a function $V \rightarrow D$, assigning a domain element to each variable in V.*

The next definitions give the interpretation of well-formed MAL-formulas.

Definition 3 *Given a signature $\Sigma = (P, F, A)$, a structure $\mathcal{S} = (S, \mathcal{I}_A, D, \mathcal{I}_P^S, \mathcal{I}_f^S)$ and an assignment $\mathcal{I}_V$, an interpretation $\mathcal{I}_t^S$ of a term t in a state s is defined as:*
$\mathcal{I}_t^S = \mathcal{I}_V(t)$ in case t is a variable
$\mathcal{I}_t^S = \mathcal{I}_f^S(f^n)(\mathcal{I}_t^S(t_1), \ldots, \mathcal{I}_t^S(t_n))$ in case t has the form $f^n(t_1, \ldots, t_n)$

Definition 4 *Given a signature $\Sigma = (P, F, A)$, a structure $\mathcal{S} = (S, \mathcal{I}_A, D, \mathcal{I}_P^S, \mathcal{I}_f^S)$ and an assignment $\mathcal{I}_V$, validity of a wff ϕ in a state s of the structure is defined as:*

$\mathcal{S}, s, \mathcal{I}_V \models \bot$	*never*	
$\mathcal{S}, s, \mathcal{I}_V \models t_1 = t_2$	*iff*	$\mathcal{I}_t^S(s)(t_1)$ *returns the same domain element as* $\mathcal{I}_t^S(s)(t_2)$
$\mathcal{S}, s, \mathcal{I}_V \models P_i^n(t_1, \ldots, t_n)$	*iff*	$(\mathcal{I}_t^S(s)(t_1), \ldots, \mathcal{I}_t^S(s)(t_n)) \in \mathcal{I}_P^S(s)(P_i^n)$
$\mathcal{S}, s, \mathcal{I}_V \models \phi \vee \psi$	*iff*	$\mathcal{S}, s, \mathcal{I}_V \models \phi$ *or* $\mathcal{S}, s, \mathcal{I}_V \models \psi$
$\mathcal{S}, s, \mathcal{I}_V \models \neg\phi$	*iff*	*not* $\mathcal{S}, s, \mathcal{I}_V \models \phi$
$\mathcal{S}, s, \mathcal{I}_V \models \exists x\, \phi(x)$	*iff*	*for some* $d \in D$ *holds* $\mathcal{S}, s, \mathcal{I}_V\{x \mapsto d\} \models \phi(x)$
$\mathcal{S}, s, \mathcal{I}_V \models \langle a \rangle \phi$	*iff*	*for some* $s' \in S$ *holds* $(s, s') \in \mathcal{I}_A(a)$ *and* $\mathcal{S}, s', \mathcal{I}_V \models \phi$
$\mathcal{S}, s, \mathcal{I}_V \models \top$	*always*	

The modal formulas $\langle a \rangle \phi$, where a is an action, mean that there is a possible occurrence of a after which ϕ holds. A formula is S-valid if it is valid in all states of a structure S. In that case, we say the structure satisfies the formula and that the structure is a model of the formula. A formula is valid if it is S-valid for every structure S.

3 Action specification in MAL

We investigate the use of MAL for writing action specifications. The first observation we make is that we do not need formulas with nested modalities for the specification of atomic actions. Nested modalities correspond to properties of sequences of actions. We assume here that all relevant system properties can be specified by focusing on individual atomic actions. We distinguish four types of MAL-formulas:

- **Conditional postcondition formulas:** $\phi \rightarrow [a]\psi$ We say that ϕ is a sufficient precondition of a with respect to the postcondition ψ. We denote sets of these formulas by Φ_{post}.
- **Guard formulas:** $\langle a \rangle \top \rightarrow \chi$ We call χ a guard of a, equivalently, a necessary precondition for the possible occurrence of a. We denote sets of these formulas by Φ_{guard}.
- **Sufficient precondition formulas:** $\sigma \rightarrow \langle a \rangle \top$ We call σ a sufficient precondition of a. We denote sets of these formulas by Φ_{suff}.
- **Static constraints:** θ These are non-modal assertions that must be obeyed under any circumstance. We denote sets of these formulas by Φ_{ic}.

The formulas ϕ, ψ, θ, σ and χ contain no modal constructs and are in disjunctive normal form.

Proposition 1 *Any MAL formula that concerns only one action symbol and contains no nested modalities, can be decomposed in a conjunction of formulas of the forms defined above.*

This shows that the above formulas are the basic ones for the specification of actions in MAL.

Proposition 2 *Each MAL-formula that is a conjunction of the above formulas is satisfiable if and only if it is satisfiable by a model with unique states (states with a unique interpretation of predicate and function symbols).*

This means that there is actually no need to distinguish states from interpretations of the predicate and function symbols. We do not have to consider different states with equal interpretations within one model or different interpretations of predicates and functions in the same state. Therefore, from now on we identify states with the interpretation of predicate and function symbols. This justifies that structures $\mathcal{S} = (S, \mathcal{I}_A, D, \mathcal{I}_P^S, \mathcal{I}_f^S)$ are from now on denoted as $\mathcal{S} = (D, S, \mathcal{I}_A)$. (Remark: this can be seen as a first step in the selection of an **intended model**; in this first step all models with equal interpretations of predicate and function symbols in different states are left out. This is actually an intuitive thing to do, because the states of the model represent states of the agent, and agent states have no duplicates either.)

Since we focus on the specification of individual atomic actions, the previous proposition should not come as a surprise. If we would have allowed the specification of properties of sequences of atomic actions, truth conditions in states influence truth conditions in more then only the immediately 'neighboring' states. This means that states with equal interpretations of predicate and function symbols can not safely be identified.

There is however one severe difficulty with specifications made with the above set of formulas: the specification can easily become inconsistent. This is one of two reasons why we suggest to drop the sufficient preconditions. The following proposition explains why.

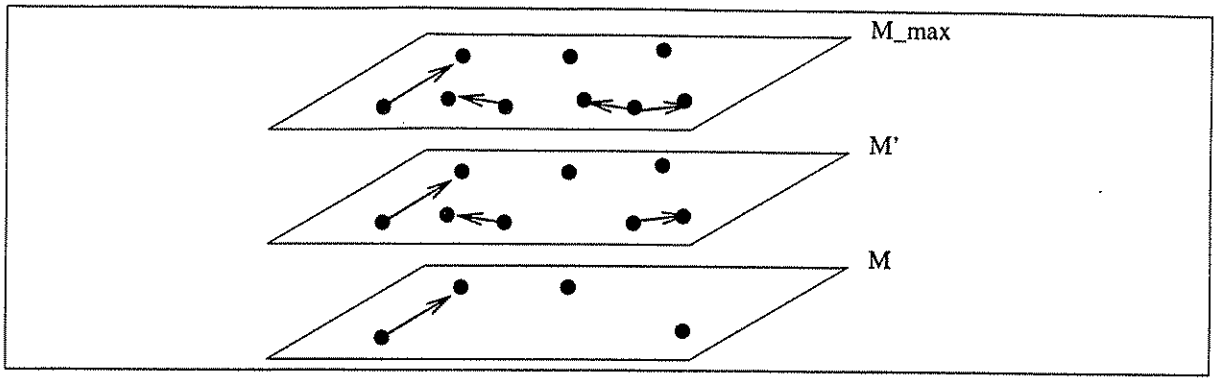


Figure 1: Comparing models on maximal reachability

Proposition 3 *Specifications that consist of conditional postcondition formulas Φ_{post}, guard formulas Φ_{guard} and static constraints Φ_{ic} are consistent if the static constraints are mutually consistent*

Also in LCM [6] the syntactic restriction that sufficient preconditions are absent is applied. Absence of this type of formulas implies transitions can always be dropped from models if they give rise to inconsistencies. Another way to deal with the problem is to weaken sufficient preconditions by specifying them as defaults [2] [17], which may be overridden.

In the next section we mention the second reason for dropping sufficient preconditions and define how the absence of sufficient preconditions formulas can be compensated for by a suitable notion of maximum reachability.

3.1 The qualification problem in MAL specifications

A common theme in the AI-literature is that it is not possible to foresee all necessary preconditions for the success of an action [8]. This problem is known as **the qualification problem**. This means that a specifier actually is never able to give a sufficient precondition for an action, which gives us a second reason to drop them from the set of specification formulas. Nevertheless we somehow need to compensate for the absence of sufficient preconditions. This is done by incorporating in the semantics the assumption that actions occur unless this contradicts guards (or static constraints, or conflicting postcondition axioms). We accomplish this by formalizing the qualification assumption in the notion of maximal reachability.

Definition 5 *Given a signature $\Sigma = (P, F, AA)$ and two structures $\mathcal{S}' = (D, S', \mathcal{I}'_{AA})$ and $\mathcal{S} = (D, S, \mathcal{I}_{AA})$*

$$\begin{aligned} &\mathcal{S}' \sqsubseteq_{mr} \mathcal{S} \text{ iff} \\ &\quad S' \subseteq S \\ &\quad \text{and} \\ &\quad \text{for all } a \in AA,\ \mathcal{I}'_{AA}(a) \subseteq \mathcal{I}_{AA}(a) \end{aligned}$$

$\sqsubseteq_{mr}$ is a partial order on structures, because $\sqsubseteq_{mr}$ can easily seen to be transitive, reflexive and anti-symmetric. The ordering just 'prefers' as much states and transitions over them as possible, thus implementing the notion of 'maximal reachability'. In Figure 2 this is reflected by the fact that all transitions and states in lower models are also in the model at the top.

Proposition 4 *Let Φ be a set of formulas for which there is a model. Then there is a $\sqsubseteq_{mr}$-maximal model.*

Preferring $\sqsubseteq_{mr}$-maximal structures leads to non-monotonic entailment. An example of this is provided by the specification $\Phi = \{[a]A(p)\}$. Under interpretation over $\sqsubseteq_{mr}$-maximal structures, $\langle a\rangle A(p)$ is entailed by Φ. However, this is no longer true for $\Phi \cup [a]\neg A(p)$.

In the next section we formalize the notion of minimal change by defining a second ordering over labeled Kripke structures.

3.2 The frame problem in MAL specifications

Postcondition formulas describe the effect of an action. However, defining the effect of an action by means of a postcondition always causes **the frame problem** [4]. This problem states that when specifying a postcondition we only want to specify conditions that change. We do not want, and often are not able, to specify all the conditions that do not change as the result of an action. So we want to make the frame assumption that everything that has not been specified to change will not change.

In the following we will define an ordering over MAL-structures that formalizes this assumption. We want the ordering to compare models on the property of 'access to "closest" possible states'. We first define a notion of distance between states.

Definition 6 *Given a structure* $\mathcal{S} = (D, S, \mathcal{I}_A)$ *and two states* $s = (\mathcal{I}_P^S, \mathcal{I}_f^S)$ *and* $s' = (\mathcal{I}'_P, \mathcal{I}'_f)$ *in* S. *The difference* $Diff(s, s')$ *between them is defined as the set of predicate instances and function values in which the states differ* ($N(V)$ *is the cardinality of a set* V*):*

$$
\begin{aligned}
&Diff(s,s') \equiv \\
&\quad \{(P_i^n, (d_1, \ldots, d_n)) \mid (d_1, \ldots, d_n) \in \mathcal{I}_P^S(P_i^n) \text{ and } (d_1, \ldots, d_n) \notin \mathcal{I}'_P(P_i^n)\} \cup \\
&\quad \{(P_i^n, (d_1, \ldots, d_n)) \mid (d_1, \ldots, d_n) \notin \mathcal{I}_P^S(P_i^n) \text{ and } (d_1, \ldots, d_n) \in \mathcal{I}'_P(P_i^n)\} \cup \\
&\quad \{(f_i^n, (d_1, \ldots, d_n)) \mid \mathcal{I}_f^S(f_i^n)(d_1, \ldots, d_n) \neq \mathcal{I}'_f(f_i^n)(d_1, \ldots, d_n)\}
\end{aligned}
$$

We want to use this measure of distance between states in the comparison between different MAL-structures. We define an ordering over Kripke structures that is connected to the definition of difference between states in such a way that structures where actions lead to 'closer' states are lower in the ordering. We will first give the two slightly different orderings, before we explain what they are really about.

Definition 7 *Given a signature* $\Sigma = (P, F, A)$ *and two structures* $\mathcal{S} = (D, S, \mathcal{I}_A)$ *and* $\mathcal{S}' = (D, S', \mathcal{I}'_A)$,

$$
\begin{aligned}
&\mathcal{S}' \sqsubseteq_{mc} \mathcal{S} \text{ iff} \\
&\quad S' = S \\
&\quad \text{and} \\
&\quad \forall a \in A,\ \forall s \in S,\ (\exists s' \in S,\ (s,s') \in \mathcal{I}'_A(a) \Leftrightarrow \exists s'' \in S,\ (s,s'') \in \mathcal{I}_A(a)) \\
&\quad \text{and} \\
&\quad \forall a \in A,\ \forall s \in S,\ \forall s' \in S,\ ((s,s') \in \mathcal{I}'_A(a) \Rightarrow \exists s'' \in S,\ ((s,s'') \in \mathcal{I}_A(a) \\
&\qquad\qquad \wedge N(Diff(s,s')) \leq N(Diff(s,s'))))
\end{aligned}
$$

$\sqsubseteq_{mc}$ is a pre-order on structures, because $\sqsubseteq_{mc}$ can easily seen to be transitive and reflexive. MC stands for minimal cardinality. A structure is called an **MC-model** of Φ if it is a $\sqsubseteq_{mc}$-minimal model of Φ. MC-models determine the minimal cardinality interpretation (semantics) of a specification (Σ, Φ).

Definition 8 *Given a signature* $\Sigma = (P, F, A)$ *and two structures* $\mathcal{S} = (D, S, \mathcal{I}_A)$ *and* $\mathcal{S}' = (D, S', \mathcal{I}'_A)$,

$$
\begin{aligned}
&\mathcal{S}' \sqsubseteq_{ms} \mathcal{S} \text{ iff} \\
&\quad S' = S \\
&\quad \text{and} \\
&\quad \forall a \in A,\ \forall s \in S,\ (\exists s' \in S,\ (s,s') \in \mathcal{I}_A(a) \Rightarrow \exists s'' \in S,\ (s,s'') \in \mathcal{I}'_A(a) \\
&\qquad\qquad \wedge (Diff(s,s') \subseteq Diff(s,s'') \vee Diff(s,s') \supseteq Diff(s,s'')))) \\
&\quad \text{and} \\
&\quad \forall a \in A,\ \forall s \in S,\ \forall s' \in S,\ ((s,s') \in \mathcal{I}'_A(a) \Rightarrow \exists s'' \in S,\ ((s,s'') \in \mathcal{I}_A(a) \\
&\qquad\qquad \wedge Diff(s,s') \subseteq Diff(s,s'')))
\end{aligned}
$$

$\sqsubseteq_{ms}$ is a pre-order on structures, because $\sqsubseteq_{ms}$ can easily seen to be transitive and reflexive. MS stands for minimal subset. A structure is called an **MS-model** of Φ if it is a $\sqsubseteq_{ms}$-minimal model of Φ. MS-models determine the minimal subset interpretation (semantics) of a specification (Σ, Φ).

The first requirement in both the $\sqsubseteq_{mc}$ and $\sqsubseteq_{ms}$ ordering is that structures can only be compared if they are based on the same set of states. In figure 1, where three models are compared on minimal change, this is reflected by the fact that all models contain the same set of black dots.

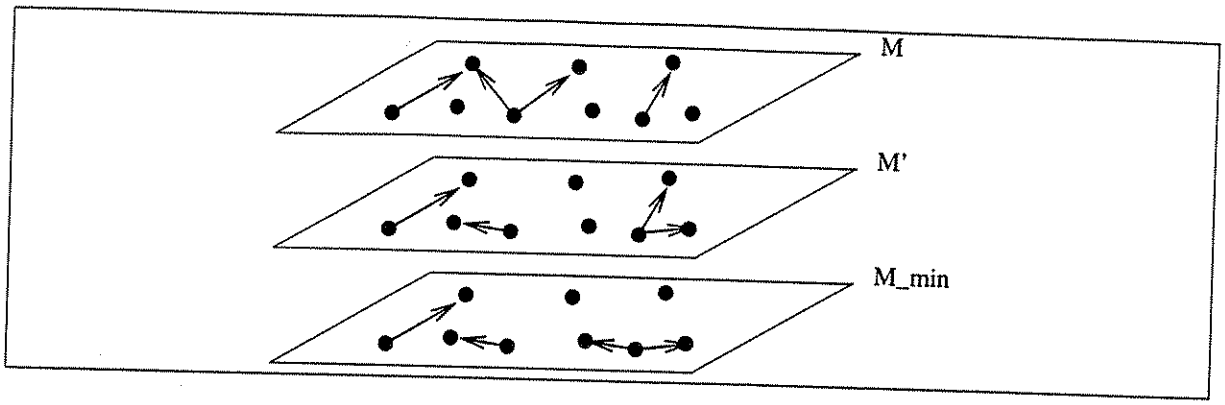


Figure 2: Comparing models on minimal change

The second requirement in both orderings forces that structures can only be compared if each transition from a state in one of the models actually corresponds to a transition from the same state in the other model, that is comparable under the minimal cardinality respectively the minimal subset criterion. For two transitions to be comparable under the minimal cardinality criterion, it is sufficient to demand that they leave from the same state; the comparison is just made on the *number* of changes that both transitions bring about. For two transitions to be comparable under the minimal subset criterion, the changes of the first must be a subset of the changes of the second or the other way around. In figure 1 this is represented by the fact that if an arrow leaves from some state in model there is also an arrow from this states in other models. (Note that the transitions may lead to different states. This is actually where we want to compare structures on; we want to prefer structures where transitions lead to "closer" states.) This is an intuitive criterion because we don't want this ordering to deal with the possible occurrence of transitions (actions); this ordering should compare structures purely on the 'length' of transitions. (This observation is also made by Brass and Lipeck [2] [17].)

The last requirement in the definition deals with this 'length' of transitions. In words it says: if $\mathcal{S}' \sqsubseteq_{mc} \mathcal{S}$ then for all transitions (s, s') in $\mathcal{S}'$ there is a transition (s, s'') in $\mathcal{S}$ that is 'longer'. In yet other words: if $\mathcal{S}' \sqsubseteq_{mc} \mathcal{S}$ then for corresponding transitions in corresponding states in the compared structures, the 'longest' transition in $\mathcal{S}'$ is always less or equal to the 'longest' transition in $\mathcal{S}$. In figure 1 the distance between dots represents the difference between states. The reason why this looks rather complicated is that we allow non-deterministic actions; we have to compare structures in which actions from one state can lead to several other states.

The difference between the semantics generated by both orderings will become more clear when we look at an example in section 3.5 and in the following section where we study minimal models for several specification classes. But we already can say that the minimal subset semantics is weaker than the minimal cardinality semantics, as is expressed by the following proposition.

Proposition 5 *An MC-model of Φ is also an MS-model of Φ.*

3.2.1 Minimal models for different specification classes

We will now investigate properties of minimal models for several classes of specification formulas. The properties we study are existence of minimal models, equivalence of the minimal cardinality and minimal subset criteria and determinism of minimal models.

We start of with the most general class we consider.

Definition 9 *We will refer to formulas Φ_{post}, Φ_{guard}, Φ_{ic} as defined before, as formulas of CLASS I*

An important property of minimal models for this most general class we consider is that if there is a model with a non-empty accessibility relation, then there is a *minimal* model with a non-empty accessibility relation.

Proposition 6 *Let Φ be a set of formulas of CLASS I that has a model $M = (D, S, \mathcal{I}_A)$ for which $\mathcal{I}_A(a) \neq \emptyset$ for all a in the formulas. Then there is an MS-model $M_{MS} = (D, S, \mathcal{I}'_A)$ of Φ for which $\mathcal{I}'_A(a) \neq \emptyset$ for all a in the formulas and an MC-model $M_{MC} = (D, S, \mathcal{I}''_A)$ of Φ for which $\mathcal{I}''_A(a) \neq \emptyset$ for all a in the formulas*

For this type of specifications MS-models and MC-models do not coincide and are not deterministic, as can be shown by a simple propositional example.

Example 1 *Consider the specification:*

$$[dial\_number](get\_connection \vee get\_busy\_tone))$$
$$get\_connection \rightarrow costs\_money$$
$$\langle dial\_number\rangle\top \rightarrow \neg costs\_money \wedge \neg get\_connection \wedge \neg get\_busy\_tone$$

Under the minimal subset criterion both the state where get(connection) holds and the state where get_busy_tone holds is reachable. Under the minimal cardinality criterion only the state where get_busy_tone holds is reachable. (Note that the constraint get_connection → costs_money is interpreted as being capable of inferring derived updates. We will come back to this in section 3.4)

We now turn our attention to quantification. Existential quantification is a source of non-determinism, as will be clear from the second class we study.

Definition 10 *the description of CLASS II*
formulas:

$$\begin{array}{ll}\Phi_{post}: & \phi \rightarrow [a]L_1 \wedge L_2 \wedge \ldots \wedge L_k \\ \Phi_{guard}: & \langle a\rangle\top \rightarrow \chi \\ \Phi_{ic}: & (M_1 \wedge M_2 \wedge \ldots \wedge M_l) \vee (N_1 \wedge N_2 \wedge \ldots \wedge N_m)\end{array}$$

with the L_h, M_i and N_j positive or negated atomic formulas (literals), and no restrictions on the quantification of variables whatsoever.

CLASS II is a subset of CLASS I. The difference is that postcondition formulas are made determinate (contain no disjunctive information) and constraints are limited to contain at maximum one disjunction in the disjunctive normal form. The following theorem holds for specifications of this type:

Proposition 7 *For formulas of CLASS II the minimal subset and minimal cardinality semantics coincide.*

Although the postcondition in the postcondition formulas of specifications of CLASS II are completely determinate, minimal models for these specifications are not deterministic. We will first define deterministic models and then give an example why minimal models for this class are not deterministic.

Definition 11 *A structure $S = (D, S, \mathcal{I}_A)$ is* ***deterministic*** *if for each state $s \in S$ and for each a there is maximally one state s' such that $(s, s') \in \mathcal{I}_A(a)$.*

Example 2 *Consider the specification:*

$$[Shoot\_a\_gun]\exists something,\ Hit(something)$$

Obviously if the domain for the Predicate Hit has several "objects", there are many possible transitions possible from a certain state. More of these transitions can be present in minimal models (the minimality criteria do not minimize nondeterminism!) This reflects the fact that there are many things that are possibly hit. In the following we will see that the existential quantification is the only source for non-determinism in this specification.

The last class we consider precisely describes the class of formulas for which minimal models are deterministic. This is an important class. Since the minimal (intended) models are deterministic for these formulas, the only intention a specifier can have when providing formulas of this form, is to specify a deterministic system. Interpreting the formulas under a non-minimal semantics would allow for non-deterministic interpretation of the formulas, which is not what is intended.

Definition 12 *the description of CLASS III*
formulas:

$$\begin{array}{ll}\Phi_{post}: & \phi \rightarrow [a]L_1 \wedge L_2 \wedge \ldots \wedge L_k \\ \Phi_{guard}: & \langle a\rangle\top \rightarrow \chi \\ \Phi_{ic}: & (M_1 \wedge M_2 \wedge \ldots \wedge M_l) \vee (N_1 \wedge N_2 \wedge \ldots \wedge N_m)\end{array}$$

with the L_h, M_i and N_j positive or negated atomic formulas (literals), and with the restriction on the quantification that variables in L_h, M_i and N_j are all universally quantified.

This class is a subset of both CLASS I and CLASS II. The difference with CLASS II is that we allow only universal quantification. We can prove the property that for formulas of CLASS III, both MS-models and MC-models are deterministic.

Proposition 8 *For a specification Spec build with formulas from CLASS III, MS-models and MC-models are deterministic.*

The former property does not hold if we take as constraints the more general class of Horn-clauses, as is shown by the next example.

Example 3

$$C(j) \rightarrow [a]A(j)$$
$$\langle a\rangle\top \rightarrow \neg A(j) \wedge \neg B(j) \wedge C(j)$$
$$A(j) \wedge C(j) \rightarrow B(j)$$

The state $\{\overline{A}(j), \overline{B}(j), C(j)\}$ has two possible follow-up states $\{A(j), B(j), C(j)\}$ and $\{A(j), \overline{B}(j), \overline{C}(j)\}$ We could of course add a distinction between base and other predicates to work around this.

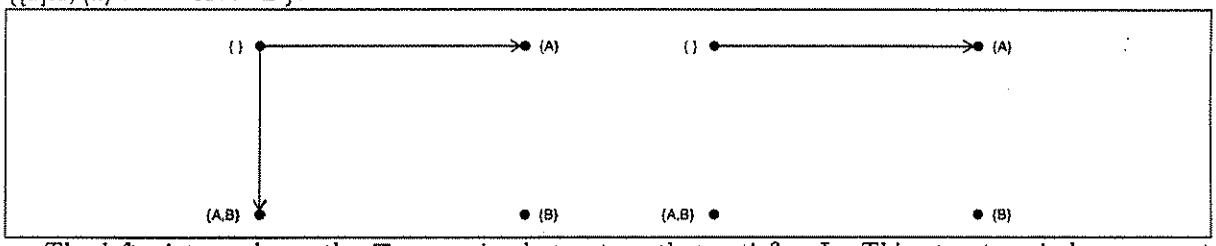
3.3 Min-Max-models

For the interpretation of specifications under both frame and qualification assumption we have to combine orderings. We do this by applying them one after the other. What, if any, is the "correct" (intuitive) order in which to do this? The answer can be found by looking at what the orderings are supposed to represent. The minimal change orderings concern the effect of actions independent from whether they occur or not. Maximal reachability deals with possible occurrence. This suggests that it is natural to apply minimal change first. First the minimal change ordering determines what actions we actually mean, by 'determining' their effects. And only after that we can 'talk' about the possible occurrence of actions. This motivates the following definition.

Definition 13 *Given a specification (Σ, Φ), a **Min-Max-model** is defined as a $\sqsubseteq_{mr}$-maximal element of the set of MS/MC-models of Φ.*

The next (propositional) example shows that defining this in reverse order leads to a not-intended model.

We take a signature with $\mathcal{P} = \{A, B\}$, and $\mathcal{A} = \{a\}$, and the following set of formulas: $\Phi = \{[a]A, \langle a\rangle\top \rightarrow \neg A \wedge \neg B\}$.



The left picture shows the $\sqsubseteq_{mr}$-maximal structure that satisfies Φ. This structure is however not MS-satisfying. This shows it is not useful to apply maximality and minimality in this order. If we start by applying minimal change, we are left with two MS-models of the example formulas, the structure with no access to other states at all and the one shown in the right picture. Of these two clearly the one in the picture is $\sqsubseteq_{mr}$-maximal and the intended one.

The MS-models of a set of formulas Φ (we mean general formulas here) form a partially ordered set under the $\sqsubseteq_{mr}$-ordering. This set is not a lattice, as is shown by the following example.

Example 4 *Take the formula $\neg(\langle a\rangle A(c) \wedge \langle a\rangle B(c))$ and the two MS-satisfying structures $\mathcal{S}$ with $\mathcal{I}_A$ is $\{\langle a, (\{\}, \{\langle A, \{d\}\rangle\})\rangle\}$ and $\mathcal{S}'$ with $\mathcal{I}'_A$ is $\{\langle a, (\{\}, \{\langle B, \{d\}\rangle\})\rangle\}$. There is no MS-satisfying structure that is an upper bound for both structures $\mathcal{S}$ and $\mathcal{S}'$. A structure that might be seen as a candidate is $\mathcal{S}''$ with $\mathcal{I}''_A$ is $\{\langle a, (\{\}, \{\langle B, \{d\}\rangle\})\rangle, \langle a, (\{\}, \{\langle A, \{d\}\rangle\})\rangle\}$, because it is an upper bound under the $\sqsubseteq_{mr}$-ordering. But this structure is not MS-satisfying (not even satisfying).*

So in general there can be more Min-Max-models of a set of formulas Φ. However, for specification formulas of the restricted form we defined, we can prove that the MS-models form a complete lattice, which leads to the following proposition.

Proposition 9 *Each specification Φ_{trans} has a* ***unique Min-Max-model*** *if it has a model.*

In Min-Max-models, the maximization in most cases causes severe non-determinism. To see this we once more look at the following example.

Example 5 *Consider the specification:*

$$[Shoot\_a\_gun]\exists something,\ Hit(something)$$

If the variable "something" ranges over an infinite domain of objects, the action Shoot_a_gun from a given state branches to an infinite number of other states.

3.4 The interpretation of constraints

We will now shortly discuss two ways to interpret constraints. The first one is already applied in the earlier examples. There Constraints have the same "status" as other specification formulas, in the sense that in the process of selecting the intended model of a specification, they are accounted for from the beginning. So, we start with all models of all specification formulas, including the constraints, apply the minimal change criterion, and then the maximal reachability criterion. This way of interpreting constraints we call the "ramification semantics" of constraints, because it is an interpretation that leads to derived effects.

There is also the possibility to postpone the role of constraints to the second step in the selection procedure of the intended model. This means we start with all models of specification formulas minus the constraints, apply the minimal change criterion, then apply the criterion that models should satisfy the constraints, and finally apply the maximal reachability criterion.

By postponing the influence of constraints to the second step in this selection procedure, many transitions are deleted. This is because it may be the case that in the first step transitions are forced to reach closest possible worlds that in the second step may be found to violate a constraint. The transitions that are "deleted" are precisely the derived updates that are present in the ramification semantics. Therefore we call this interpretation of constraints simply the "constraint semantics".

3.5 Example

The following example specification is inspired by the Yale shooting problem:

$$\begin{array}{l} Gun\_loaded\_sharp \rightarrow [fire\_gun]Gun\_blown\_up \vee Bullet\_emitted \\ Gun\_loaded\_blank \rightarrow [fire\_gun]Gun\_blown\_up \vee Air\_and\_dust\_emitted \\ \rightarrow [fire\_gun]Big\_noise \\ \langle fire\_gun\rangle\top \rightarrow \neg Gun\_blown\_up \\ \langle fire\_gun\rangle\top \rightarrow Gun\_loaded\_sharp \vee Gun\_loaded\_blank \\ Bullet\_emitted \rightarrow Something\_is\_hit \\ \neg(Gun\_loaded\_sharp \wedge Gun\_loaded\_blank) \end{array}$$

First we conclude that the ramification semantics is the most intuitive interpretation for the constraints. Under the constraint semantics the condition *Something_is_hit* can actually never change to true as a result of the action *fire_gun*. It can only be true after the action *fire_gun* if it was already true in the state before *fire_gun* took place.

We will also have to choose between the the minimal subset and the minimal cardinality criterion. The choice is not too difficult if we look at the postcondition $Gun\_blown\_up \vee Bullet\_emitted$. Given the fact that we interpret constraints as possibly leading to ramifications, the truth of *Bullet_emitted* will imply the truth of *Something_is_hit* in resulting states. This means that the action *fire_gun* usually 'has the choice' between making one atom true (*Gun_blown_up*) or two (*Bullet_emitted* and *Something_is_hit*). The minimal cardinality semantics will choose the first alternative which is really counterintuitive. The minimal subset semantics just makes no choice; both successor states are possible. This means that the minimal subset semantics is a more non-deterministic interpretation.

It is easy to give examples of how the action *fire_gun* can be qualified. Adding $\{Gun\_blown\_up\}$ or $\{\neg Big\_noise\}$ would leave us with no transitions at all. Adding $\{\neg Gun\_loaded\_sharp\}$ would result in transitions that never make *Something_is_hit* true.

Finally we want to point out that the language to 'query' the intended model can be any language whose semantics can be defined using labeled Kripke structures, as is also argued by Lifschits e.a. [15]. As

an example of this we give some properties stated in Dynamic Logic using process operators like iteration test and sequence, that might be checked on the intended model of the example formulas.

$Gun\_loaded\_blank \rightarrow [fire\_gun^*]Gun\_loaded\_blank$

It says that if the gun was loaded with a blank it will stay loaded with blanks after a possibly infinite amount times of shooting.

$[Gun\_loaded\_sharp?\ ;\ fire\_gun](\neg Gun\_blown\_up \rightarrow Something\_is\_hit)$.

It says that if the gun is fired in a situation where it is loaded with a bullet, and as the result of it the gun will not blow up, then something is hit.

4 Comparison with other work

The frame problem and the associated problems of qualification and ramification are common themes in the AI literature on knowledge representation and reasoning about action and change [4].

Ryan [20] argues that structure consisting of a hierarchies of agents and sub-agents can be used to generate frame axioms automatically. The structure, that must be provided by the specifier, induces a notion of locality. This makes it possible to automatically generate two kinds of completion axioms: "local attribute axioms" stating that local attributes can only be affected by local actions and "local action axioms" stating that local actions can only affect local attributes. This last axiom involves a second order property, because it refers to attributes and not to their values. The proposed completion is however not complete. The first form of incompleteness is that within agents, all local attributes can still be affected by all local actions. The second form is that throughout the whole structure of agents all non-local attributes can still be affected by all non-local actions. Whether this gives rise to unintended results depends on the agent structure given by the specifier.

Reiter's approach to the frame problem [19] is to rewrite "effect axioms" to "successor state axioms". Effect axioms say for each action which predicates change their value if the action is performed. Successor state axioms say for each predicate which actions change when performed. Reiter's approach consists mainly of a change of focus from specific actions to specific predicates and a form of completion on them. An important difference with our approach is that Reiter uses sufficient conditions for the possibility of an action a, while we use only necessary conditions. So Reiter can actually never forbid actions from being possible in certain situations, he can only force them to be possible. As argued, we chose not to specify sufficient conditions, because this might cause problems in the presence of static constraints, which is exactly the problem Reiter et al. run into [16]. It is easy to show that Reiter's "successor state axioms", that are generated from "effect axioms" by performing a form of completion, can be expressed in MAL. However, Reiter's approach has several limitations. First of all it should be mentioned that in Reiter's approach the completion can only be performed on postcondition formulas that are determinate (no disjunctive postconditions). This means that this approach does not deal with non-deterministically specified effects. Second, the completion is not possible in the presence of static constraints that possibly contradict the effect. Third, the completion is not "complete". For details on this we refer to a full version of this paper [3].

Borgida et al. [1] take the perspective of the designer of specification languages and discuss ways to state that "nothing else changes" by syntactic as well as semantic means. Our work can be regarded as introducing a richer semantics for the specification language to capture this.

Giunchiglia, Kartha and Lifschitz introduce the action language $\mathcal{AR}$ [9], which is an extension of the language $\mathcal{A}$ [7], introduced by Gelfond and Lifschitz. $\mathcal{AR}$ differs from $\mathcal{A}$ in that it also deals with ramifications. There is a straightforward translation of most elements of the language $\mathcal{AR}$ into elements of (the propositional version of) our language. Under this translation their (minimal) interpretation function Res_D perfectly matches our Min-Max-models (because they only consider determinate postconditions there is no difference between the minimal subset or minimal cardinality criterion). The difference between their language and ours is that they can express that an effect possibly occurs (A **possibly changes** F **if** P) which can not be expressed in our language. On the other hand we can express that the effect of an action is a choice between two or more alterations ($[a](A \vee B)$), which can not be expressed in their language. Interesting is that both constructs are claimed to represent the non-deterministic aspect of actions.

Winslett's work on database update semantics [24] focuses on a model-oriented approach to updates, de-emphasizing the relation between the specification and the models. Instead, we base our semantics on the declarative semantics of a specification in MAL, which allows us to reason about updates in the same language.

Brass and Lipeck [2] [17] study action specification with the help of defaults. They also define orderings over modal interpretations. Frame and other assumptions are represented by formulas interpreted as defaults. This still puts the responsibility on the specifier to provide such formulas, which is not always desirable. Furthermore their models represent "action traces" and do not allow for non-determinism.

5 Conclusions and future work

In this paper we defined a preferential semantics for an important class of first order MAL formulas, that reflects the principles of minimal change and maximal reachability. Several results concerning more refined classes of specification formulas were reported.

We plan to compare existing completion procedures, such as the one suggested by Reiter, with our semantics, and to compare existing procedures for scenario generation and reachability analysis with our semantics. We also plan to investigate ways to generate the intended model (Min-Max-model) from a given specification. For that we will have to limit ourselves to finite domains and a finite number of actions. We will have to find a suitable representation for models and an algorithm that connects this representation to specifications. This opens possibilities for code generation and preferential model checking (to be read as "checking preferential models") using a model-checker like SMV [5] [18]. An example of the kind of properties that could be checked by preferential model checking was given at the end of the example section.

Another interesting future research area is the study of the notions of minimal change and maximal reachability in the context of concurrent actions and processes.

References

[1] A. Borgida, J. Mylopoulos, and R. Reiter. On the frame problem in procedure specifications. *IEEE Transactions on Software Engineering*, 21:785–798, 1995.

[2] S. Brass, U. W. Lipeck, and P. Resende. Specification of object behaviour with defaults. In G. Koschorreck U. W. Lipeck, editor, *Proceedings of the International Workshop on Information Systems – Correctness and Reusability (IS-CORE'93)*, pages 155–177. Informatik-Berichte 01/93, Universit t Hannover, Januari 1993.

[3] J.M. Broersen and R.J. Wieringa. Minimal semantics for action specifications in first-order dynamic logic. Technical Report IR-439, Faculty of Mathematics and Computer Science, Vrije Universiteit, Amsterdam, November 1997.

[4] Frank M. Brown, editor. *The frame problem in artificial intelligence: proceedings of the 1987 workshop, April 12-15, 1987, Lawrence, Kansas.* Kaufmann, 1987.

[5] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2), April 1986.

[6] R.B. Feenstra and R.J. Wieringa. LCM 3.0: a language for describing conceptual models. Technical Report IR-344, Faculty of Mathematics and Computer Science, Vrije Universiteit, Amsterdam, December 1993.

[7] Michael Gelfond and Vladimir Lifschitz. Representing action and change by logic programs. *Journal of Logic Programming*, 17(2/3&4):301–321, 1993.

[8] Matthew L. Ginsberg and David E. Smith. Reasoning about action 2: the qualification problem. *Artificial Intelligence*, 35:311–342, 1988.

[9] E. Giunchiglia, G. N. Kartha, and V. Lifschitz. Actions with indirect effects (extended abstract). In *Proc. AAAI Spring Symposium'95 on Extending Theories of Actions*, 1995.

[10] F. Golshani, T.S.E. Maibaum, and M.R. Sadler. A modal system of algebras for database specification and query/update support. In *Proceedings of the Nineth International Conference on Very Large Databases*, pages 331–359, 1983.

[11] D. Harel. *First Order Dynamic Logic.* Springer, 1979. Lecture Notes in Computer Science 68.

[12] P. Jeremaes, S. Khosla, and T.S.E. Maibaum. A modal (action) logic for requirements specification. In D. Barnes and P. Brown, editors, *Software Engineering 86*, pages 278–294. Peter Peregrinus Ltd., 1986.

[13] S. Khosla and T.S.E. Maibaum. The prescription and description of state based systems. In B. Banieqbal, H. Barringer, and A. Pnueli, editors, *Temporal Logic in Specification*, pages 243–294. Springer, 1987. Lecture Notes in Computer Science 398.

[14] S. Khosla, T.S.E. Maibaum, and M. Sadler. Database specification. In T.B. Jr. Steel and R. Meersman, editors, *Database Semantics (DS-1)*, pages 141–158. North-Holland, 1986.

[15] V. Lifschitz. Two components of an action language. In *Annals of Mathematics and Artificial Intelligence.* 1997. to appear.

[16] F. Lin and R. Reiter. State Constraints Revisited. *Journal of Logic and Computation*, 4(5):655–678, 1994. Special Issue on Action and Processes.

[17] U. W. Lipeck and S. Brass. Object-oriented system specification using defaults. In H. Marburger K. von Luck, editor, *Management and Processing of Complex Data Structures - Third Workshop on Information Systems and Artificial Intelligence 1994*, Lecture Notes in Computer Science 777, pages 22–43. Springer-Verlag, 1994.

[18] K.L. McMillan. *Symbolic Model Checking.* Kluwer Academic Publishers, 1993.

[19] R. Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In Vladimir Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: P apers in Honor of John McCarthy.* Academic Press, 1991.

[20] M. Ryan, J. Fiadeiro, and T. Maibaum. Sharing actions and attributes in modal action logic. In T. Ito and A.R. Meyer, editors, *Theoretical Aspects of Computer Software*, pages 569–593. Springer, 1991. Lecture Notes in Computer Science 526.

[21] R.J. Wieringa, J.-J. Ch. Meyer, and H. Weigand. Specifying dynamic and deontic integrity constraints. *Data and Knowledge Engineering*, 4:157–189, 1989.

[22] R.J. Wieringa and J.-J.Ch. Meyer. Actor-oriented specification of dynamic and deontic integrity constraints. In B. Talheim, J. Demetrovics, and H.-D. Gerhardt, editors, *3rd Symposium om Mathematical Fundamentals of Database and Knowledge Base Systems (MFDBS 91)*, pages 89–103. Springer, 1991. Lecture Notes in Computer Science 495. ftp://ftp.cs.vu.nl/pub/roelw/91-Actors.ps.Z. ftp://ftp.cs.vu.nl/pub/roelw/91-Actors.ps.Z.

[23] R.J. Wieringa, H. Weigand, J.-J. Ch. Meyer, and F. Dignum. The inheritance of dynamic and deontic integrity constraints. *Annals of Mathematics and Artificial Intelligence*, 3:393–428, 1991. ftp://ftp.cs.vu.nl/pub/roelw/91-Inheritance.ps.Z. ftp://ftp.cs.vu.nl/pub/roelw/91-Inheritance.ps.Z.

[24] M. Winslett. *Updating Logical Databases.* Cambridge University Press, 1990.

An Argumentative Framework for Reasoning with Inconsistent and Incomplete Information

Alejandro J. García[1] and Guillermo R. Simari[1] and Carlos I. Chesñevar[1]

Abstract. We present here a knowledge representation language, where defeasible and non-defeasible rules can be expressed. The language has two different negations: *classical negation*, which is represented by the symbol "∼" used for representing contradictory knowledge; and *negation as failure*, represented by the symbol "not" used for representing incomplete information. Defeasible reasoning is done using a argumentation formalism. Thus, systems for acting in a dynamic domain, that properly handle contradictory and/or incomplete information can be developed with this language.

An *argument* is used as a defeasible reason for supporting conclusions. A conclusion q will be considered justified only when the argument that supports it becomes a *justification*. Building a justification involves the construction of a non-defeated argument $\mathcal{A}$ for q. In order to establish that $\mathcal{A}$ is a non-defeated argument, the system looks for *counterarguments* that could be *defeaters* for $\mathcal{A}$. Since defeaters are arguments, there may exist defeaters for the defeaters, and so on, thus requiring a complete dialectical analysis. The system also detects, avoids, circular argumentation. The language was implemented using an abstract machine defined and developed as an extension of the Warren Abstract Machine (WAM).

1 The language

Our language is defined in terms of two types of program clauses:

- *extended program clauses* [2](EPC): l <- $q_1, \ldots, q_n$.
- *defeasible program clauses*[3](DPC): l -< $q_1, \ldots, q_n$.

There are two different negations: *classical negation*, which is represented by the symbol "∼" used for representing contradictory knowledge; and *negation as failure*, represented by the symbol "not" used for representing incomplete information. In both kinds of clauses l is a literal (*i.e.*, a predicate "p" or a negated predicate "$\sim p$"), and each q_i ($n \geq 0$) is a literal, or a literal preceded by the symbol not. Thus, classical negation is allowed in the consequent of a clause, and negation as failure over literals is allowed in the antecedent. If $n = 0$, an EPC becomes "$l \leftarrow$ true." (or simply "l.") and is called a *fact*, whereas a DPC becomes "$l \prec$ true.", and is called a *presumption*.

We will use the usual PROLOG typographic conventions for an EPC, except that we will write "head <- body" rather than "head :- body"; and "head -< body" for a defeasible clause. An EPC is used to represent sound (*i.e.*, not defeasible) information such as: bird(X) <- penguin(X) which expresses that "all penguins are birds", whereas a DPC is used to represent defeasible knowledge such as: fly(X) -< bird(X) which expresses that "presumably, a bird can fly" or "usually, a bird can fly."

As mentioned earlier, program clauses can contain two types of negation. Classical negation (∼) can be used in clauses such as:

```
~guilty(X) <- innocent(X).
~free(X) -< ~innocent(X).
```

to express that "an innocent is not guilty", and that "usually, if someone is not innocent, then it is not free." Negation as failure (not) may also be used in clauses such as:

```
innocent(X) -< not guilty(X).
~cross-railway-tracks -< not ~train-is-coming.
```

to express that "assume that someone is innocent whenever it has not been proven that s/he is guilty" and "generally, do not cross railroad tracks if it cannot be proven that no train is coming." This kind of rules could not be written with only one type of negation.

Operationally, the difference between the two negations is as follows: a query "∼q" succeeds when there exists a proof for "∼q." On the other hand a query "not q" will succeed when no proof can be found for "q". In fact, proving a negated literal "∼p" is carried out just as if the "∼" symbol were syntactically part of the predicate name, thereby treating "∼p" as an atomic predicate name. In this system, a proof is a formal derivation called an *argument*, which *justifies* a query q. Arguments and justifications will be introduced in the following section.

With two types of negations, the Closed World Assumption (CWA) of a predicate p could be expressed within the language, writing the clause

```
~p(X) <- not p(X).
```

i.e., "~p(X) will be derived whenever the proof of p(X) fails." Also, new forms of CWA can be written with the obvious

[1] Artificial Intelligence Research Group – Departamento de Ciencias de la Computación – Universidad Nacional del Sur – Av.Alem 1253 – (8000) Bahía Blanca, ARGENTINA – e-mail: { ccgarcia, grs, ccchesne }@criba.edu.ar

[2] We use this terminology following [8] because we allow classical negation as done there. However, the system presented here, can accommodate inconsistency whereas the language reported in [8] cannot.

[3] Given the similarity in use and syntax, we use the term 'clause' for this construction, even though it is not properly a clause, but a meta relation between the head and body of the rule.

interpretation:

```
p(X) <- not ~p(X).
p(X) <- not p(X).
~p(X) <- not ~p(X).
```

Nevertheless, if a defeasible clause is used, a defeasible notion of CWA could be represented:

```
~p(X) -< not p(X).
```

which expresses that "the failure of the proof of `p(X)` is a good reason to assume `~p(X)`". Thus, the following clauses can be written:

```
~dead(X) -< not dead(X).
dangerous(X) -< not ~dangerous(X).
```

Definition 1.1 *A defeasible logic program (DLP) is a finite set of EPCs and DPCs.*

Let $\mathcal{P}$ be a DLP; then, we will distinguish the subset $\mathcal{S}$ of EPC in $\mathcal{P}$, and the subset $\mathcal{D}$ of DPC in $\mathcal{P}$.

Example 1.1 : Here follows a DLP that will be referred to in other examples:

```
fly(X) -< bird(X).
~fly(X) -< chicken(X).
fly(X) -< chicken(X),scared(X).
chicken(coco) -< true.
penguin(petete) -< true.
~dangerous(X) -< pet(X).
dangerous(X) -< tiger(X).
bird(X) <- chicken(X).
scared(coco).
bird(X) <- penguin(X).
~fly(X) <- penguin(X).
pet(kitty).
tiger(kitty).
```

□

Given a DLP $\mathcal{P}$, a *defeasible derivation* for a query " `-< q`" is a finite set of EPC and DPC obtained by backward chaining from `q` as in a PROLOG program, using both strict and defeasible rules in the order specified by the DLP. The symbol "$\sim$" is considered as part of the predicate when generating a defeasible derivation.

Example 1.2 : Using the DLP of example 1.1, there are defeasible derivations for each of the following queries:

```
-< ~fly(petete)
-< fly(petete)
-< fly(coco)
-< ~fly(coco)
```

□

It can be seen from Example 1.2, that the defeasible derivation notion does not forbid inferring two complementary literals from a given DLP $\mathcal{P}$. In order to allow only one of two complementary goals to be accepted as a sensible possibility, we need a criterion for choosing between the two.

In the next section the notion of *argument* will be introduced to allow the defeasible argumentation formalism developed in [20, 19] which will allow us to define an inference scheme for the language. But first, we need to explicate when a set of clauses is deemed consistent.

Definition 1.2 (Consistency) *A set of program clauses $\mathcal{A}$ is* consistent *if there is no defeasible derivation for any pair of complementary literals (with respect to classical negation "$\sim$"). Conversely, a set of program clauses $\mathcal{A}$ is* inconsistent *if there are defeasible derivations for a pair of complementary literals.*

Given a DLP $\mathcal{P}$, the set $\mathcal{S}$ must be consistent, although the set $\mathcal{D}$, and hence $\mathcal{P}$ itself (*i.e.*, $\mathcal{S} \cup \mathcal{D}$), may be inconsistent. It is only in this form that a DLP may contain potentially inconsistent information.

Example 1.3 : Here follows a set of rules that supports inconsistent conclusions, because `fly(petete)` and `~fly(petete)` can be derived.

```
penguin(petete) -< true.    bird(X) <- penguin(X).
fly(X) -< bird(X).          ~fly(X) <- penguin(X).
```

Observe also, that the DLP of example 1.1 is an inconsistent set of program clauses, although its associated set $\mathcal{S}$ is a consistent one. □

2 Arguments, Rebuttals and Defeaters

In this framework, answers to queries must be supported by *arguments*. However, arguments may be defeated by other arguments. A query `q` will succeed if the supporting argument for it is not defeated; it then becomes a *justification*. Before defining formally the notion of justification, we define arguments, counterarguments and defeaters.

Definition 2.1 (Argument) *An* argument $\mathcal{A}$ *for a query h, also denoted $\langle \mathcal{A}, h \rangle$, is a subset of* ground instances *of* DPCs *of $\mathcal{D}$, such that: (1) There exists a defeasible derivation for h from $\mathcal{S} \cup \mathcal{A}$, (2) $\mathcal{S} \cup \mathcal{A}$ is consistent, and (3) $\mathcal{A}$ is minimal with respect to set inclusion.*

The above definition is adapted from [20] to fit in this framework. It states that an argument is a consistent defeasible derivation for a given query h, using a minimal set of rules. Note that EPCs are not part of the argument.

Example 2.1 : Consider the DLP of example 1.1.
The query `-< ~fly(coco)` has the argument:

$$\mathcal{A}_1 = \left\{ \begin{array}{l} \texttt{~fly(coco) -< chicken(coco).} \\ \texttt{chicken(coco) -< true.} \end{array} \right\}$$

However, the query `-< fly(coco)` has two arguments:

$$\mathcal{A}_2 = \left\{ \begin{array}{l} \texttt{fly(coco) -< bird(coco).} \\ \texttt{chicken(coco) -< true.} \end{array} \right\}$$

$$\mathcal{A}_3 = \left\{ \begin{array}{l} \texttt{fly(coco) -< chicken(coco),scared(coco).} \\ \texttt{chicken(coco) -< true.} \end{array} \right\}$$

The query -< ~fly(petete) has the argument:

$$\mathcal{A}_4 = \{ \texttt{penguin(petete) -< true.} \}$$

Finally, there is no argument for -< fly(petete) because its defeasible derivation is inconsistent with respect to $\mathcal{S}$ (see example 1.3). □

Definition 2.2 (Sub-argument) *An argument $\langle \mathcal{B}, q \rangle$ is a* sub-argument *of $\langle \mathcal{A}, h \rangle$ if $\mathcal{B} \subseteq \mathcal{A}$.*

Definition 2.3 (Counterargument or rebuttal) *We say that $\langle \mathcal{A}_1, h_1 \rangle$* counterargues or rebuts *$\langle \mathcal{A}_2, h_2 \rangle$ at literal h, if and only if there exists a sub-argument $\langle \mathcal{A}, h \rangle$ of $\langle \mathcal{A}_2, h_2 \rangle$ such that the set $\mathcal{S} \cup \{h_1, h\}$ is inconsistent.*

An argument is in fact a *proof tree*, involving rules from $\mathcal{S}$ and $\mathcal{D}$. Hence, arguments will be depicted as triangles abstracting a tree shape [19]. The upper vertex of the triangle will be labeled with the argument's conclusion, and the argument will be associated with the triangle itself. Sub-arguments will be represented as smaller triangles inside a big one, which corresponds to the main argument at issue. Figure 1 shows the graphical representation of an argument $\langle \mathcal{A}_2, h_2 \rangle$ with one sub-argument $\langle \mathcal{A}, h \rangle$, and a counterargument $\langle \mathcal{A}_1, h_1 \rangle$.

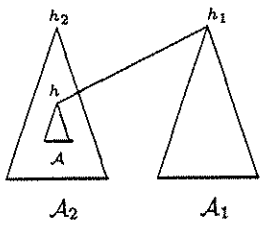


Figure 1. Argument $\langle \mathcal{A}_1, h_1 \rangle$ counterargues $\langle \mathcal{A}_2, h_2 \rangle$ at h

Example 2.2 : Continuing with Example 2.1, the argument $\mathcal{A}_1$ is a counterargument for both $\mathcal{A}_2$ and $\mathcal{A}_3$ (at fly(coco)), and also $\mathcal{A}_2$ and $\mathcal{A}_3$ are counterarguments for $\mathcal{A}_1$ (both at ~fly(coco)). Note that the argument $\mathcal{A}_4$ has no counterarguments. □

As we mentioned before, the justification process for proving q involves the construction of a non-defeated argument $\mathcal{A}$ for q. In order to verify whether an argument is non-defeated, its associated counterarguments $\mathcal{B}_1, \mathcal{B}_2, \ldots \mathcal{B}_k$ are examined, each of them being a potential (defeasible) reason for rejecting $\mathcal{A}$. If any $\mathcal{B}_i$ is better than (or unrelated to) $\mathcal{A}$, then $\mathcal{B}_i$ is a candidate for defeating $\mathcal{A}$. If the argument $\mathcal{A}$ is better than $\mathcal{B}_i$ then $\mathcal{B}_i$ is not taken into account.

So we must clarify what makes an argument "better" than an other one. In this work, as a particular example, we will define a formal criterion called *specificity* which allows to discriminate between two conflicting arguments. However, the notion of *defeating argument* can be formulated independently of which particular argument-comparison criterion is used. Namely, if some $\mathcal{B}_i$ is better (*i.e.*, more specific, in our case) than $\mathcal{A}$, then $\mathcal{B}_i$ is called a *proper defeater* for $\mathcal{A}$; if neither argument is better than the other, a blocking situation occurs, and we will say that $\mathcal{B}_i$ is a *blocking defeater* for $\mathcal{A}$. This is an skeptical criterion that could be changed producing a different formal decision procedure.

Definition 2.4 (Defeating argument)
An argument $\langle \mathcal{A}_1, h_1 \rangle$ defeats *an argument $\langle \mathcal{A}_2, h_2 \rangle$ at literal h, if and only if there exists a sub-argument $\langle \mathcal{A}, h \rangle$ of $\langle \mathcal{A}_2, h_2 \rangle$ such that $\langle \mathcal{A}_1, h_1 \rangle$ counterargues $\langle \mathcal{A}_2, h_2 \rangle$ at h, and either:*
(1) $\langle \mathcal{A}_1, h_1 \rangle$ is strictly more specific than $\langle \mathcal{A}, h \rangle$ (then $\langle \mathcal{A}_1, h_1 \rangle$ is a proper *defeater of $\langle \mathcal{A}_2, h_2 \rangle$); or*
(2) $\langle \mathcal{A}_1, h_1 \rangle$ is unrelated by specificity to $\langle \mathcal{A}, h \rangle$ (then $\langle \mathcal{A}_1, h_1 \rangle$ is a blocking *defeater of $\langle \mathcal{A}_2, h_2 \rangle$).*

The next definition characterizes specificity as defined in [14, 19] (adapted to fit in this framework). Intuitively, this notion of specificity favors two aspects in an argument: it favors an argument (1) with more information content and (2) with shorter derivations. In other words, an argument is deemed better that another if it is *more precise* or *more concise*. This notion is made formally precise in the next definition. We use the symbol "$\mid\!\sim$ " to denote a defeasible derivation (*i.e.*, $\mathcal{P} \mid\!\sim\ h$ means that h has a defeasible derivation from $\mathcal{P}$), and the symbol "$\vdash$" to denote a derivation where only EPCs are used. Let $\mathcal{S}_G$ be the maximal subset of $\mathcal{S}$ that does not contain facts. Let $\mathcal{F}$ be the set of literals in $\mathcal{P}$ that have a defeasible derivation.

Definition 2.5 (Specificity)
An argument $\mathcal{A}_1$ for h_1 is strictly more specific than *an argument $\mathcal{A}_2$ for h_2 (denoted $\langle \mathcal{A}_1, h_1 \rangle \succ \langle \mathcal{A}_2, h_2 \rangle$) if and only if:*
(1) For all $G \subseteq \mathcal{F}$: if $\mathcal{S}_G \cup G \cup \mathcal{A}_1 \mid\!\sim\ h_1$ and $\mathcal{S}_G \cup G \not\mid\!\sim\ h_1$, then $\mathcal{S}_G \cup G \cup \mathcal{A}_2 \mid\!\sim\ h_2$.
(2) There exists $G' \subseteq \mathcal{F}$ such that $\mathcal{S}_G \cup G' \cup \mathcal{A}_2 \mid\!\sim\ h_2$ and $\mathcal{S}_G \cup G' \not\mid\!\sim\ h_2$ and $\mathcal{S}_G \cup G' \cup \mathcal{A}_1 \not\mid\!\sim\ h_1$.

Example 2.3 : Continuing with Example 2.1, argument $\mathcal{A}_1$ is strictly more specific than the argument $\mathcal{A}_2$ because $\mathcal{A}_1$ does not use the EPC bird(X) <- chicken(X) and therefore is a more direct argument. However, argument $\mathcal{A}_3$ is strictly more specific than $\mathcal{A}_1$, because it contains more information. Then, $\mathcal{A}_1$ is a proper defeater for $\mathcal{A}_2$, and $\mathcal{A}_3$ is a proper defeater for $\mathcal{A}_1$. □

3 Dialectical Trees and Justifications

Since defeaters are arguments, there may exist defeaters for defeaters, and so on. This means that it is necessary to pursue argument supports to ascertain their well-foundedness. This justification process is called a *dialectical analysis*. It can be specified in the context of Logic Programming in the following way (here \+ stands for PROLOG's negation as failure):

```
justify(Q) :- find_argument(Q,A), \+ defeated(A)
defeated(A) :- find_defeater(A,D), \+ defeated(D)
```

The above description leads in a natural way to the use of trees to organize our dialectical analysis. In order to accept an argument $\mathcal{A}$ as a justification for q, a tree structure can be generated. The root of the tree will correspond to the

argument $\mathcal{A}$ and every inner node will represent a defeater (proper or blocking) of its father. Leaves in this tree will correspond to non-defeated arguments. This structure is called a *dialectical tree.*

Definition 3.1 (Dialectical tree) *[19]*
Let $\mathcal{A}$ be an argument for h. A dialectical tree *for $\langle \mathcal{A}, h \rangle$, denoted $T_{\langle \mathcal{A}, h \rangle}$, is recursively defined as follows:*
(1) A single node labeled with an argument $\langle \mathcal{A}, h \rangle$ with no defeaters (proper or blocking) is by itself a dialectical tree for $\langle \mathcal{A}, h \rangle$. This node is also the root of the tree.
(2) Suppose that $\langle \mathcal{A}, h \rangle$ is an argument with defeaters (proper or blocking) $\langle \mathcal{A}_1, h_1 \rangle$, $\langle \mathcal{A}_2, h_2 \rangle$, ..., $\langle \mathcal{A}_n, h_n \rangle$. We construct the dialectical tree for $\langle \mathcal{A}, h \rangle$, $T_{\langle \mathcal{A}, h \rangle}$, by labeling the root node of with $\langle \mathcal{A}, h \rangle$ and by making this node the parent node of the roots of the dialectic trees for $\langle \mathcal{A}_1, h_1 \rangle$, $\langle \mathcal{A}_2, h_2 \rangle$, ..., $\langle \mathcal{A}_n, h_n \rangle$, i.e., $T_{\langle \mathcal{A}_1, h_1 \rangle}$, $T_{\langle \mathcal{A}_2, h_2 \rangle}$, ..., $T_{\langle \mathcal{A}_n, h_n \rangle}$.

Nodes in the dialectical tree can be recursively marked as *defeated* or *undefeated* nodes (D-nodes and U-nodes respectively). Let $\mathcal{A}$ be an argument for a literal h, and $T_{\langle \mathcal{A}, h \rangle}$ be its associated dialectical tree.

Definition 3.2 (Marking of a dialectical tree)
(1) Leaves of $T_{\langle \mathcal{A}, h \rangle}$ are U-nodes.
(2) Let $\langle \mathcal{B}, q \rangle$ be an inner node of $T_{\langle \mathcal{A}, h \rangle}$. Then $\langle \mathcal{B}, q \rangle$ will be an U-node *iff every child of $\langle \mathcal{B}, q \rangle$ is a* D-node. *The node $\langle \mathcal{B}, q \rangle$ will be a* D-node *iff it has at least an* U-node *as a child.*

This definition suggests a bottom-up marking procedure, through which we are able to determine if the root of a dialectical tree turns out to be marked as defeated or undefeated.

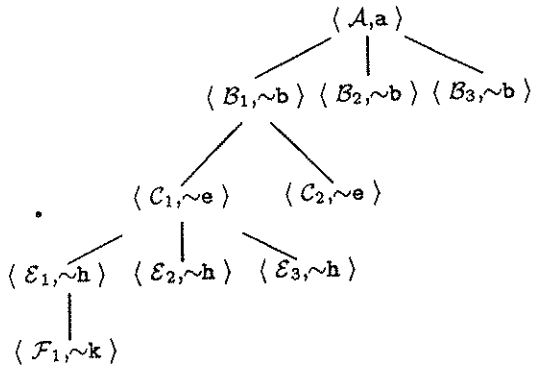


Figure 2. Dialectical tree for example 3.1

Example 3.1 : Consider the following DLP:

```
a -< b.      ~e -< l.     c ← true.
b -< c.      ~h -< k.     f ← true.
~b -< e.     ~b -< i.     i ← true.
e -< f.      k -< l.      l ← true.
~b -< c,f.   ~h -< c.     n ← true.
~e -< h.     ~h -< l.
h -< i.      ~k -< n,l.
```

Here the argument $\mathcal{A}$=`{ a -< b. b -< c. }` for a can be built. Argument $\mathcal{A}$ has three defeaters attacking the literal b: $\mathcal{B}_1$= `{ ~b -< e. e -< f. }` $\mathcal{B}_2$= `{ ~b -< c,f. }` and $\mathcal{B}_3$= `{~b -< i. }`. $\mathcal{B}_2$ is a proper defeater for $\mathcal{A}$, the other two are blocking defeaters. Argument $\mathcal{B}_1$ has also two blocking defeaters attacking the literal e: $\mathcal{C}_1$= `{~e -< h. h -< i.}` and $\mathcal{C}_2$= `{ ~e -< l.}`. Argument $\mathcal{C}_1$ has three blocking defeaters: $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{E}_3$, and finally $\mathcal{E}_1$has one proper defeater $\mathcal{F}_1$. The complete dialectical tree is shown in Figure 2. Figure 3 shows the same dialectical tree after applying the marking procedure of Definition 3.2. Note that nodes labeled with "#" need not to be considered in the analysis and pruning of the tree can be done. □

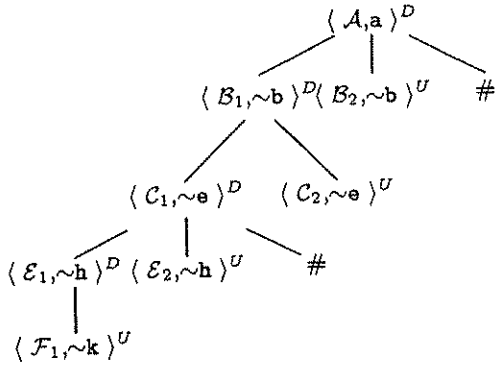


Figure 3. Marked dialectical tree for Figure 2 with pruning

Definition 3.3 (Justification) *Let $\mathcal{A}$ be an argument for a literal h, and let $T_{\langle \mathcal{A}, h \rangle}$ be its associated dialectical tree. The argument $\mathcal{A}$ will be a justification for a literal h if the root of $T_{\langle \mathcal{A}, h \rangle}$ is an U-node.*

If a query h has a justification, then it is considered 'justified', and the answer to the query will be YES. Nevertheless, there are several reasons for an argument not to be a justification: there may exist a non-defeated proper defeater, or a non-defeated blocking defeater, or there may be no argument at all. Therefore, in a DLP there are four possible answers for a query " —≺ h":

- YES, if there is a justification $\mathcal{A}$ for h.
- NO, if for each possible argument $\mathcal{A}$ for h, there exists at least one proper defeater for $\mathcal{A}$ marked as U-node.
- UNDECIDED, if for each possible argument $\mathcal{A}$ for h, there are no proper defeaters for $\mathcal{A}$ marked U-node, but there exists at least one blocking defeater for $\mathcal{A}$ marked U-node.
- UNKNOWN, if there exists no argument for h.

Finally, an example that shows all the concepts defined above is presented.

Example 3.2 : Given the DLP of Example 1.1, the following arguments can be built:

$\mathcal{A}_1$= { ~fly(coco) -< chicken(coco). ; chicken(coco) -< true. }
for ~fly(coco);

$\mathcal{A}_2$= { fly(coco) -< bird(coco). ; chicken(coco) -< true. }
for fly(coco);

$\mathcal{A}_3$= { fly(coco) -< chicken(coco),scared(coco). ; chicken(coco) -< true. }
for fly(coco);

$\mathcal{B}_1$= { ~dangerous(kitty) -< pet(kitty). }
for ~dangerous(kitty);

$\mathcal{B}_2$= { dangerous(kitty) -< tiger(kitty). }
for dangerous(kitty); and

$\mathcal{C}$={ penguin(petete) -< true. }
for ~fly(petete)

If the query -< fly(coco) is submitted, the system first builds the argument $\mathcal{A}_2$ and looks for a defeater for it. Then, defeater $\mathcal{A}_1$ is found, so the system tries to find a defeater for $\mathcal{A}_1$, and then $\mathcal{A}_3$ is found. Since $\mathcal{A}_3$ has no defeaters, it becomes an U-node. Therefore $\mathcal{A}_1$ becomes a D-node and $\mathcal{A}_2$ becomes an U-node. Thus, $\mathcal{A}_2$ *is* a justification for fly(coco), and the answer for this query is YES.

The query "~fly(coco)" is answered NO, because the argument $\mathcal{A}_1$ has the proper (non-defeated) defeater $\mathcal{A}_3$. The argument $\mathcal{B}_1$ has the blocking defeater $\mathcal{B}_2$, since $\mathcal{B}_2$ has no defeaters, then the query "dangerous(kitty)" is UNDECIDED. Note that the query "~dangerous(kitty)" is also UNDECIDED. Finally the answer for "fly(petete)" is UNKNOWN because there is no argument for this query. □

4 Negation as Failure

As mentioned earlier, the language has two different negations: classical negation, which is represented by the symbol "~" used for representing contradictory knowledge; and negation as failure, represented by the symbol "not" used for representing incomplete information. Operationally, the difference between the two negations is as follows: a query "~q" succeeds when there exists a justification for "~q." On the other hand a query "not q" will succeed when no justification for 'q" can be found. In our language, negation satisfies the coherent principle established in [2]: "If ~p succeeds then not p also succeeds."

Example 4.1 : Consider the following DLP:

```
p(X) -< not q(X).
q(a).
q(X) -< r(X).
r(b).
r(c).
~q(X) -< r(X),s(X).
s(c).
```

Here the query "~q(c)" succeeds because it has a justification, however, there is no justification for "~q(b)" (actually there is no argument for this query). The query "not q(d)" succeeds because there is no justification for q(d). However, since there is a justification for q(b), then the query "not q(b)" fails. Note that the query "not q(c)" succeeds, because although there is an argument for q(c), there is a defeater for it, so there is no justification for q(c). Thus, the queries p(c) and p(d) succeed, whereas p(a) and p(b) fail. □

5 Avoiding circular argumentation

A DLP is a finite set of program clauses, and therefore there is a finite number of arguments that may be involved in a dialectical tree. Nevertheless, we need to impose conditions in order to avoid cycles in this tree. In [19], it is shown that circular argumentation is a particular case of *fallacious argumentation*. There, a detailed analysis exposes different kinds of undesirable situations, and solutions are proposed accordingly. We next briefly present the problems and their solutions and refer to [19] for details.

In order to analyze fallacious argumentation, it is useful to see a dialectical tree as a set of *argumentation lines*. Following [19], this notion is formally defined as follows. Let $\langle \mathcal{A}_0, h_0 \rangle$ be an argument, and let $\mathcal{T}_{\langle \mathcal{A}_0, h_0 \rangle}$ be its associated dialectical tree.

Definition 5.1 (Argumentation line) *Every path λ from the root $\langle \mathcal{A}_0, h_0 \rangle$ to a leaf $\langle \mathcal{A}_n, h_n \rangle$ in $\mathcal{T}_{\langle \mathcal{A}_0, h_0 \rangle}$, denoted $\lambda = [\langle \mathcal{A}_0, h_0 \rangle, \langle \mathcal{A}_1, h_1 \rangle, \langle \mathcal{A}_2, h_2 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle]$, is called an* argumentation line *for h_0.*

In each argumentation line $\lambda = [\langle \mathcal{A}_0, h_0 \rangle, \langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_i, h_i \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle]$, the argument $\langle \mathcal{A}_0, h_0 \rangle$ is supporting the main query h_0, and every argument $\langle \mathcal{A}_i, h_i \rangle$ defeats its predecessor $\langle \mathcal{A}_{i-1}, h_{i-1} \rangle$. Therefore, for $k \geq 0$, $\langle \mathcal{A}_{2k}, h_{2k} \rangle$ is a *supporting* argument for h_0 and $\langle \mathcal{A}_{2k+1}, h_{2k+1} \rangle$ is an *interfering* argument for h_0. In other words, every argument in the line either supports h_0's justification or interferes with it. Therefore, an argumentation line can be split in two disjoint sets: λ_S of supporting arguments, and λ_I of interfering arguments. Thus, an argumentation line λ can be construed as an alternate sequence of supporting and interfering arguments as in any ordered debate. In a dialectical tree, there are as many argumentation lines as leaves in the tree, and each can end up in a supporting or an interfering argument.

We next present three types of fallacious argumentation, and establish necessary and sufficient conditions to avert them.

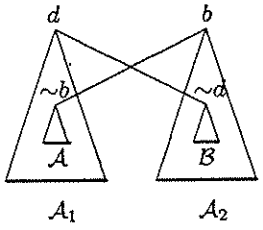


Figure 4. Reciprocal defeaters

The first problematic situation is shown in Figure 4. This happens when a pair of arguments defeat each other. In this

case, $\langle \mathcal{A}_1, d \rangle$ defeats $\langle \mathcal{A}_2, b \rangle$, attacking the subargument $\langle \mathcal{B}, \sim d \rangle$, but $\langle \mathcal{A}_2, b \rangle$ also defeats $\langle \mathcal{A}_1, d \rangle$ attacking the subargument $\langle \mathcal{A}, \sim b \rangle$. Clearly, this situation is nonsensical as it leads to an infinite argumentation line. The first condition expressed in Definition 5.3 prevents this situation.

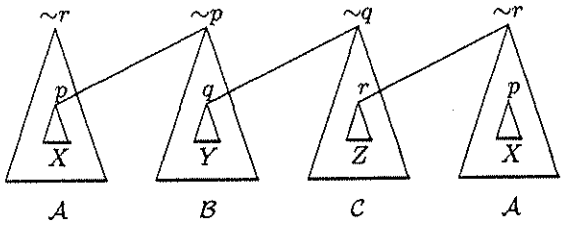


Figure 5. Contradictory argumentation line

Figure 5 shows a case where the same argument ($\mathcal{A}$) becomes both a supporting and an interfering argument of itself. This too is a nonsensical situation as it arises because the supporting argument $\mathcal{C}$ has a subargument Z for the literal r, which is contradictory with the purpose of arguing in favor of $\sim r$ (argument $\mathcal{A}$). An argument like $\mathcal{C}$ ought to be avoided in a sound argumentation line. Clearly, there should be agreement among supporting arguments (respectively interfering) in any argumentation line. This underlies the second condition in Definition 5.3. This is expressed formally based on the notion of argument *concordance* as proposed in [19] and recalled next. Supporting (respectively interfering) arguments should be mutually *concordant* so as to make the dialectical process of argumentation coherent.

Definition 5.2 (Concordance)
Two arguments $\langle \mathcal{A}_1, h_1 \rangle$ and $\langle \mathcal{A}_2, h_2 \rangle$ are concordant *iff the set $\mathcal{S} \cup \mathcal{A}_1 \cup \mathcal{A}_2$ is consistent. More generally, a set of arguments $\{\langle \mathcal{A}_i, h_i \rangle\}_{i=1}^{n}$ is concordant iff $\mathcal{S} \cup \bigcup_{i=1}^{n} \mathcal{A}_i$ is consistent.*

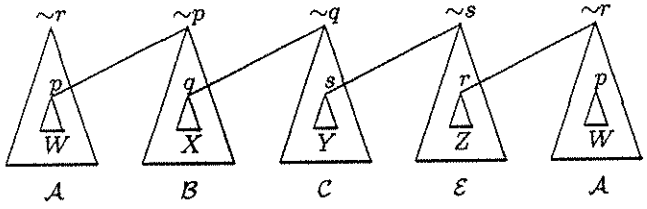


Figure 6. Circular argumentation

Finally, Figure 6 shows an example of circular argumentation, where the same argument $\mathcal{A}$ is introduced again in the line as a supporting argument. This is avoided by the third condition in Definition 5.3, which disallows the more general problematic situation where a subargument of an earlier argument is reintroduced further down the argumentation line.

These three situations are averted by requiring that all argumentation be addressed be *acceptable* as defined below:

Definition 5.3 (Acceptable argumentation line) *[19] Let $\lambda = [\langle \mathcal{A}_0, h_0 \rangle, \langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_i, h_i \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle]$ be an argumentation line, λ is an* acceptable argumentation line *iff:*
(1) For every defeater $\langle \mathcal{A}_i, h_i \rangle$ and every proper subargument $\langle \mathcal{B}, q \rangle$ of $\langle \mathcal{A}_i, h_i \rangle$, $\langle \mathcal{B}, q \rangle$ and $\langle \mathcal{A}_{i-1}, h_{i-1} \rangle$ are concordant; that is, $\mathcal{S} \cup \{q, h_{i-1}\}$ must be consistent.
(2) The sets λ_S of supporting arguments, and λ_I of interfering arguments of λ must each be concordant sets of arguments.
(3) No argument $\langle \mathcal{A}_k, h_k \rangle$ in λ is a subargument of an earlier argument $\langle \mathcal{A}_i, h_i \rangle$ of λ $(i < k)$.

Hence, with these conditions averting undesirable situations, an *acceptable dialectical tree* is a dialectical tree where every argumentation line is acceptable. Then the notion of justification can be properly defined as follows [19].

Definition 5.4 (Justification) *The argument $\langle \mathcal{A}, h \rangle$ is a justification for h iff its associated dialectical tree is acceptable and the root node of $\mathcal{T}_{\langle \mathcal{A}, h \rangle}$ is a* U*-node.*

6 Implementation

In order to develop an efficient defeasible argumentation system, an abstract machine called JAM (Justification Abstract Machine) [7] has been designed as an extension of the Warren's abstract machine (WAM) [1]. The JAM architecture has an instruction set, a memory structure and a set of registers for building arguments, counterarguments, and in this form obtaining justifications for queries. A compiler for defeasible logic programs was developed. It takes a defeasible logic program as its input and produces a sequence of JAM instructions as its output. The JAM was built as a virtual machine, and an interpreter for defeasible logic programs was developed over this machine.

7 Related Work

Other formalisms for defeasible argumentation have been separately developed. In [6] P. Dung has proposed a very abstract and general argument-based framework, where he completely abstracts from the notions of argument and defeat. In contrast we have defined an object language for representing knowledge and a concrete notion of argument and defeat, Dung's approach assumes the existence of a set of arguments ordered by a binary relation of defeat. However, he defines various notions of 'argument extensions', which are intended to capture various types of defeasible consequence.

Inspired by legal reasoning, H. Prakken and G. Sartor [17, 18] have developed an argumentation system that like us, uses the language of extended logic programming. They introduce a *dialectical proof theory* for an argumentation framework fitting the abstract format developed by Dung, Kowalski *et al.* [6, 4]. However, since they are inspired by legal reasoning, the protocol for dispute is rather different from our dialectical approach. A proof of a formula takes the form of a *dialogue tree*, where each branch of the tree is a dialogue between a *proponent* and an *opponent*. Proponent and opponent have different rules for introducing arguments, leading to an asymmetric dialogue. Later, Prakken [16] generalized the system to default logic's language.

R. Kowalski and F. Toni [9] have outlined a formal theory of argumentation, in which defeasibility is stated in terms of

non-provability claims. They argue that defeasible reasoning with rules of the form P if Q can be understood as "exact" reasoning with rules of the form P if Q and S cannot be shown, where S stands for one or more defeasible "non-provability claims".

Other related works are those by Vreesvijk [21], Bondarenko [3], Pollock [13], Loui [10], and Nute [11, 12]. The interested reader is referred to the following surveys in defeasible argumentation: Prakken & Vreesvijk [15], and Chesñevar *et al.* [5].

8 Conclusions

We have presented a knowledge representation language, that uses an argumentation formalism for defeasible reasoning, Defeasible rules allow to represent tentative knowledge, but also strong rules can be used. Since classical negation and negation as failure are both available in the language, contradictory and incomplete information can be represented. Several forms of CWA can be represented directly within the language.

Conclusions are supported by arguments, but when contradictory information is reached during a derivation, the defeasible argumentation formalism provides a way for deciding between competing arguments. If new information arises, then new arguments could be constructed and previous conclusions could be withdrawn. Thus, a correct behavior for a dynamic domain is obtained.

In order to develop efficient defeasible systems, the language was implemented using an abstract machine defined and implemented as an extension of the Warren Abstract Machine (WAM).

Acknowledgments

Alejandro J. García wishes to thank especially Hassan Aït-Kaci for many helpful discussions and suggestions. The authors are also grateful to Henry Prakken for helpful comments, and the anonymous referees for their suggestions. This work was partially supported by CONICET and Secretaría de Ciencia y Técnica UNS.

REFERENCES

[1] H. Aït-Kaci, *Warren's Abstract Machine—A Tutorial Reconstruction*, MIT Press, 1991.

[2] José Alferes and Luis M. Pereira, 'Contradiction: when avoidance equals removal (part i and ii)', in *Proc. of Ext. of Logic Programming, 4th International Workshop ELP'93 St. Andrews U.K*, (March 1993).

[3] A. Bondarenko, P.M. Dung, R.A. Kowalski, and F. Toni, 'An abstract, argumentation-theoretic approach to default reasoning', *Artificial Intelligence*, **93**, 63–101, (1997).

[4] A. Bondarenko, F. Toni, and R.A. Kowalski, 'An assumption-based framework for non-monotonic reasoning', *Proc. 2nd. International Workshop on Logic Programming and Non-monotonic Reasoning*, 171–189, (1993).

[5] C. I. Chesñevar, A. Maguitman, and R.P.Loui, 'Logical models of arguments', *submitted to ACM Computing Surveys*, (1998).

[6] Phan M. Dung, 'On the acceptability of arguments and its fundamental role in nonmonotonic reasoning and logic programming and n-person games', *Artificial Intelligence*, **77**, 321–357, (1995).

[7] Alejandro J. García, *Defeasible Logic Programming: Definition and Implementation* (MSc Thesis), Departamento de Cs. de la Computación, Universidad Nacional del Sur, Bahía Blanca, Argentina, July 1997.

[8] M. Gelfond and V. Lifschitz, 'Logic programs with classical negation', in *Proceedings of the 7th International Conference on Logic Programming*, eds., D. Warren and P. Szeredi, pp. 579–597. MIT Press, (1990).

[9] Robert A. Kowalski and Francesca Toni, 'Abstract argumentation', *Artificial Intelligence and Law*, **4**(3-4), 275–296, (1996).

[10] Ronald P. Loui, Jeff Norman, Joe Altepeter, Dan Pinkard, Dan Craven, Jessica Lindsay, and Mark Foltz, 'Progress on room 5: A testbed for public interactive semi-formal legal argumentation', in *Proc. of the 6th. International Conference on Artifcial Intelligence and Law*, (July 1997).

[11] D. Nute, 'Basic defeasible logic', in *Intensional Logics for Programming*, ed., Luis Fariñas del Cerro, Clarendon Press, Oxford, (1992).

[12] D. Nute, 'Defeasible logic', in *Handbook of Logic in Artificial Intelligence and Logic Programming, Vol 3, Nonmonotonic Reasoning and Uncertain Reasoning*, eds., C.J. Hogger D.M. Gabbay and J.A.Robinson, 355–395, Oxford University Press, (1994).

[13] John L. Pollock, *Cognitive Carpentry: A Blueprint for How to Build a Person*, Massachusetts Institute of Technology, 1995.

[14] David L. Poole, 'On the Comparison of Theories: Preferring the Most Specific Explanation', in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 144–147. IJCAI, (1985).

[15] H. Prakken and G. Vreesvijk, 'Logical systems for defeasible argumentation (to appear)', in *Handbook of Philosophical Logic, second edition*, ed., Gabbay, (1998).

[16] Henry Prakken, *Logical Tools for Modelling Legal Argument. A Study of Defeasible Reasoning in Law*, Kluwer Law and Philosophy Library, 1997.

[17] Henry Prakken and Giovanni Sartor, 'A system for defeasible argumentation, with defeasible priorities', in *Proc. of the International Conference on Formal Aspects of Practical Reasoning, Bonn, Germany*. Springer Verlag, (1996).

[18] Henry Prakken and Giovanni Sartor, 'Argument-based logic programming with defeasible priorities', *Journal of Applied Non-classical Logics*, **7**(25-75), (1997).

[19] Guillermo R. Simari, Carlos I. Chesñevar, and Alejandro J. García, 'The role of dialectics in defeasible argumentation', in *Anales de la XIV Conferencia Internacional de la Sociedad Chilena para Ciencias de la Computación*. Universidad de Concepción, Concepción (Chile), (November 1994).

[20] Guillermo R. Simari and Ronald P. Loui, 'A Mathematical Treatment of Defeasible Reasoning and its Implementation', *Artificial Intelligence*, **53**, 125–157, (1992).

[21] Gerard A.W. Vreeswijk, 'Abstract argumentation systems', *Artificial Intelligence*, **90**, 225–279, (1997).

Formal Semantics of the Core of AGENT-0

Koen V. Hindriks, Frank S. de Boer, Wiebe van der Hoek and John-Jules Meyer

University Utrecht, Department of Computer Science
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
{koenh,frankb,wiebe,jj}@cs.ruu.nl

Abstract

AGENT-0 is a rule-based, agent-oriented programming (AOP) language. An agent is an entity with a complex mental state, consisting of beliefs and commitments. The properties of the mental state of agents built into the language are specified by a modal logic. The link between the modal logic and the programming language AGENT-0, however, is unclear. Moreover, the logic does not specify how the mental states of agents change in time. In this paper, we provide a clear and formal semantics which specifies both the static and dynamic aspects of (part of) the mental states of AGENT-0 agents. We use a two-layered approach by separating the semantic systems for specifying the semantics of the basic language constructs and the interpreter for the language. Our approach yields a better insight into the use of rules, and allows for a detailed comparison with other agent programming languages. The approach is a first step towards the design of a logic that is based on a formal semantics which defines AGENT-0 agents.

1 Introduction

AGENT-0 is an experimental programming language to program intelligent agents. An intelligent agent in AGENT-0 is an entity with a mental state, consisting of the beliefs and commitments of the agent, that is capable of interacting with its environment and deciding what to do. Rules provide the basic means for decision-making and updating the mental state. Programming AGENT-0 agents means building programs that specify how the mental states of agents change. According to Thomas [6],"agent programs are transition functions on these mental states; given an agent's current mental state and input, the transition function specifies the agent's new mental state and output."

The emphasis on mental states in agent programming makes it imperative to state precisely and explicitly what such a state is. This is one of the reasons for the requirement of a formal semantics of mental state as part of an agent programming paradigm in [5]. In [5, 6] a modal logic is used to define several of the mental components. The formal semantics of this logic, however, contrasts with the lack of a formal semantics of the agent programming language itself. There is no clear link between the modal logic and the language AGENT-0. Moreover, the logic does not define how the mental states of agents change. It therefore does not provide us with any insight into the decision-making of agents and the rules which the AGENT-0 agents use for this purpose.

A formal semantics of the agent programming language is needed for several reasons. First of all, we need it to be able to formally reason about such agents, and to understand

what the language constructs mean. In particular, a formal semantics will provide a first step towards the design of a logic for reasoning about AGENT-0 agents. It also provides a formal definition of the rule-based decision-making of agents. Secondly, we need a formal semantics to compare AGENT-0 with other agent languages. The formal specification of AGENT-0 we provide highlights an interesting difference in decision-making of AGENT-0 agents compared with, for example, AgentSpeak(L) ([4]) and 3APL ([2, 1]) agents. And finally, a formal semantics may prevent ad hoc implementations of agent programming languages.

2 The Core of AGENT-0

An AGENT-0 agent has a number of features which provide such agents with capabilities to interact with its environment, to decide what to do, and to update its mental state. These features include the following: communication, time, capabilities, simple, conditional and refrain actions, beliefs, commitments, and commitment rules.

As a first approximation, we construct a semantics for a subset of AGENT-0. We call this subset the *core* of AGENT-0 and define it to be the language without multi-agent features like communication or reference to other agents, and without reference to time. Thus, the core of AGENT-0 as we define it includes beliefs, capabilities, three types of action, commitments, and rules. The specification of the formal semantics for this core is based on the informal explanation of AGENT-0 in [5]. Our strategy for defining the semantics is to separate the semantic systems specifying the meaning of the basic constructs of the programming language like beliefs and rules and the interpreter for the language. This strategy is based on our research into our own agent language 3APL (cf. [2, 1]).

2.1 Syntax

Before we can give a definition of the syntax of actions, we need to define terms and a language for expressing belief. A term is a constant or a variable. We do not allow variables ranging over agents, beliefs, and action statements. Given a set P of predicate symbols, a set C of constants, and a set Var of variables, terms T and atomic formulae P are constructed inductively by: (i) $\mathsf{Var} \subseteq \mathsf{T}$, (ii) $\mathsf{C} \subseteq \mathsf{T}$, and (iii) if $P \in \mathsf{P}$ of arity n, and $t_1, \ldots, t_n \in \mathsf{T}$, then $P(t_1, \ldots, t_n) \in \mathsf{At}$.

Mental state formulae and actions are defined by simultaneous induction. Mental state formulae are used to express both the beliefs of the agent and its commitments to action. Negated action formulae express that there is no commitment to perform the action.

When communication is left out of AGENT-0, three types of actions remain: (i) *simple actions* of the form `(DO <privateaction>)`, constructed from a given set of so-called *private actions*, (ii) *conditional actions* of the form `(IF <mntlcond> <action>)`, where `<mntlcond>` expresses a condition on the mental state of the agent, and (iii) *refrain actions* of the form `(REFRAIN <action>)`. The refrain action `(REFRAIN <action>)` is a type of action that precludes commitment to actions of the form `<action>`. In the following definition, we recapitulate the previous outline of the action types in AGENT-0 in a somewhat different notation.

Definition 2.1 Let B be a set of action symbols. Then the set of actions A, the set of mental state literals Lit, and the mental state language $\mathcal{L}^m$ are inductively defined by:

- if $a \in \mathsf{B}$ of arity n, and $t_1, \ldots, t_n \in \mathsf{T}$, then $a(t_1, \ldots, t_n) \subseteq \mathsf{A}$, called *private actions*, (The set of private actions is also denoted by $\mathcal{P}$),

- if $a \in \mathsf{A}$ and $\phi \in \mathcal{L}^m$, then $(\phi : a) \in \mathsf{A}$, called *conditional actions*,
- if $a \in \mathsf{A}$, then $\delta a \in \mathsf{A}$, called *refrain actions*.
- if $\phi \in \mathsf{At}$, then $\phi, \neg\phi \in \mathsf{Lit}$, and if $a \in \mathsf{A}$, then $a, \neg a \in \mathsf{Lit}$.
- $\mathsf{Lit} \subseteq \mathcal{L}^m$, and if $\phi, \psi \in \mathcal{L}^m$, then $\phi \wedge \psi, \phi \vee \psi \in \mathcal{L}^m$.

In AGENT-0 rules are of the condition-action type. These rules determine under which conditions an action should be taken into consideration. When the multi-agent features are left out, rules are of the form: `(COMMIT <mntlcond> <action>*)`. In our notation we write this as:

Definition 2.2 The set of *commitment rules* R is defined by: if $\phi \in \mathcal{L}^m$, and $\Pi \subseteq \mathsf{A}$, then $\leftarrow \phi \mid \Pi \in \mathsf{R}$.

The syntax of commitment rules is explained by the fact that condition-action rules can be viewed as a subset of more general rules of the form $\Pi \leftarrow \phi \mid \Pi'$. The notation $\leftarrow \phi \mid \Pi'$ highlights that condition-action rules are rules of this more general format, where the head $\Pi = \emptyset$. Note, however, that the condition part of the rules in AGENT-0 may refer to both beliefs and committed actions. Nevertheless, these rules do not *modify* any existing commitments when fired.

Syntactically, an agent program is a set of capabilities, a set of initial beliefs, and a set of commitment rules. Capabilities are pairs consisting of a condition and a private action. The (mental state) condition is a precondition that specifies under what circumstances an action is assumed to succeed when executed.

Definition 2.3 An *agent program* is a tuple $\langle \mathsf{Cap}, \sigma_0, \Gamma \rangle$, where Cap is a set of *capabilities*, i.e. a set of actions of the form $(\phi : a)$, $\sigma_0 \subseteq \mathsf{At}$ is the set of *initial beliefs*, and $\Gamma \subseteq \mathsf{R}$ is a set of *commitment rules*.

2.2 Semantics

The notion of mental state is the basic concept in agent-oriented programming. As cited in the introduction, agent programs can be viewed as transition functions on mental state. We use transition systems to define the transitions of the mental states of agents which occur as a consequence of either executing actions or applying rules.

Definition 2.4 A *mental state* is a pair $\langle \Pi, \sigma \rangle$, where $\Pi \subseteq \mathsf{A}$ is a set of actions, also called *commitments*, and σ is a set of beliefs.

In the next sections, the *operational semantics* of the core of AGENT-0 is defined by *two* Plotkin-style transition systems ([3]). Formally, a transition system is a deductive system which allows to *derive* the transitions of a program. A transition system consists of a set of *transition rules* that specify the meaning of each programming construct in the language. Such a rule consists of some premises and a conclusion. A transition system defines a transition relation $\longrightarrow$ on mental states. The transition rules transform these mental states.

The two transition systems defined specify the dynamics of the mental states of agents. We use a two-layered approach. In this section, we define the meaning of the basic language

constructs. In section 3, we define the semantics of the main control loop of the interpreter of AGENT-0.

To be able to specify the semantics of actions, we need a specification of the effect of private actions. In [5] the meaning of private actions is not discussed in detail. However, a number of remarks suggest that private actions should be taken as updates on the set of beliefs of the agent. This is the view we will take here. For this purpose, we introduce the (partial) function $\mathcal{T} : \mathcal{P} \times \wp(\mathsf{At}) \rightarrow \wp(\mathsf{At})$ which specifies what type of update is performed by a private action.

Definition 2.5

$$\frac{\mathcal{T}(a, \sigma) = \sigma'}{\langle \{\ldots, a, \ldots\}, \sigma \rangle \longrightarrow \langle \{\ldots\}, \sigma' \rangle}$$

A conditional action is executed by testing its condition and if the test succeeds by committing to the action, i.e. incorporating the action in the set of commitments. Variables in the test may also retrieve data from the belief database and set of commitments. The values retrieved are recorded in a substitution θ. The consequence relation $\models$ is assumed to be given (and intuitively clear, [5]).

Definition 2.6 Let θ be a substitution, and $\Pi = \{\ldots, (\phi : a), \ldots\}$.

$$\frac{\Pi, \sigma \models \phi\theta}{\langle \{\ldots, (\phi : a), \ldots\}, \sigma \rangle \longrightarrow \langle \{\ldots, a\theta, \ldots\}, \sigma \rangle}$$

A refrain action δa removes actions of the form a from the set of commitments. In this way, it prevents the execution of these actions. It is not clear from [5] if and when a refrain action itself is removed from the set of commitments. We have chosen to not delete the action after executing it, since this type of action is probably most often used for *safety reasons*. I.e., for example, to prevent destructive behaviour removing certain opportunities or to prevent certain undesirable effects of the action which is to be refrained from.

Definition 2.7

$$\frac{\delta a \in \Pi}{\langle \Pi, \sigma \rangle \longrightarrow \langle \Pi \setminus \{a\theta \mid \theta \text{ a substitution}\}, \sigma \rangle}$$

In AGENT-0 there are no operators for constructing complex actions, for example, by sequencing a number of simple actions. This lack of constructs for programming control flow is one of the things which suggests that AGENT-0 supports a *bottom-up approach*. By a *bottom-up approach* we mean here that in contrast with a *top-down approach* the agent does not decide what to do next by fixing a high-level goal, but decides what to do next by looking only at the circumstances the agent finds itself in. This lack of goal-driven behaviour of agents has been one of the reasons to extend the language with such features in the language PLACA [6].

Another feature which also suggests AGENT-0 supports a bottom-up approach is the type of rules allowed to program an agent. The rules to program agents are condition-action rules. We mean by this that the rules do not *modify* any existing (high-level) goals of the agent by substituting plans for achieving them, but just *add* new commitments to the set

of commitments if the conditions of a rule are satisfied. In this sense we could say that a language like AGENT-0 is *rule-driven*, while languages like AgentSpeak(L) ([4]) and 3APL ([2, 1]) are *goal-driven*.

The difference between the two approaches is built into the corresponding control structures of the interpreters. The difference corresponds to the different *practical syllogisms* used for decision-making in the goal-driven and rule-driven interpreters:

Practical Syllogism corresponding to the Goal-Driven Approach (PSG):
If (P1) the agent intends to achieve a goal g, and (P2) believes that g will not be achieved unless *some* plan p from a set P of plans will be executed, then (C) the agent intends to execute a plan p from P (according to some selection criteria),

Practical Syllogism corresponding to the Rule-Driven Approach (PSR):
If (P1) the agent believes he is in situation S_Π in which he has made commitments Π, and (P2) believes S_Π makes action a obligatory, then (C) the agent intends to perform action a.

An explanation of the PSG syllogism is to view it as a reasoning scheme which may be used by the agent to achieve a goal by means of *some* plan. The PSR syllogism is best explained as a reasoning scheme to guarantee the commitment to *all* actions of a particular form. Thus, the first might profitably be used to infer a *possible* means to achieve a goal, while the second is more suited to be used as a means to infer the *necessity* to perform an action, i.e. to guarantee that some actions are performed in certain circumstances. The two approaches therefore are dual approaches and correspond to the duality of the possibility and necessity modalities. For this reason, it is interesting to note that in [5] the concept of *obligation* is taken as basic instead of that of *motivation*. The commitment rules in AGENT-0 code the (conditional) obligations of agents. In [5] Shoham actually somewhat overstates, we think, the contrast between the two different modes of decision making. According to him, the decision-making in AGENT-0 "reflects absolutely no motivation of the agent, and merely described the actions to which the agent is obligated." (p. 67) The tools for programming an agent in AGENT-0 on the one hand, and AgentSpeak(L) and 3APL on the other hand, thus are derived from two different perspectives on decision-making.

In AGENT-0 *all* applicable instances of a rule are fired. I.e., a rule is fired for every match of the condition of the rule with the belief database and commitments of the agent. The choice to fire all applicable rules corresponds to the PSR syllogism. We introduce some notation to express this type of rule application express more succinctly. Let Θ be a set of substitutions, and Π be a set of actions. Then: (i) $\Pi\theta = \{a\theta \mid a \in \Pi\}$, and (ii) $\Pi\Theta = \bigcup_{\theta\in\Theta} \Pi\theta$.

Formally, a rule fires for each substitution that satisfies the condition of the rule in the current mental state. It should be noted that a variant of the rule is used to prevent any undesired interference with variables in the rule and the belief database and commitments (cf. also [2]).

Definition 2.8 Let Θ be the set of *all* substitutions θ such that $\Pi, \sigma \models \phi\theta$.

$$\frac{\leftarrow \phi \mid \Pi' \in' \Gamma}{\langle \Pi, \sigma\rangle \longrightarrow \langle \Pi \cup \Pi'\Theta, \sigma\rangle}$$

where $\leftarrow \phi \mid \Pi' \in' \Gamma$ means that $\leftarrow \phi \mid \Pi'$ is a variant of a rule in Γ such that no variables in $\leftarrow \phi \mid \Pi'$ occur in Π or σ.

Although we did not formally define a transition rule for general rules of the form $\pi \leftarrow \phi \mid \pi'$, but focused on the rules most prominent in AGENT-0, the more general rules will be convenient when we define an interpreter for AGENT-0. We will therefore use these more general rules later on, and refer the reader to [2] for details. π in a rule $\pi \leftarrow \phi \mid \pi'$ is called the *head* of the rule, ϕ the *guard* or *condition* of the rule, and π' the *body* of the rule. The head and the body of a rule may be empty, which is denoted by $\Box$. A rule $\Box \leftarrow \phi \mid \Pi$ may also be substituted by a set of rules $\Box \leftarrow \phi \mid \pi$ for all $\pi \in \Pi$; this set yields the same result when all rules in the set are fired as when the single rule $\Box \leftarrow \phi \mid \Pi$ is fired (cf. section 3).

In the following definition we define some shorthand notation used in the remainder of the paper. It is assumed, without any loss of generality that all rules are of the form $\pi \leftarrow \phi \mid \pi'$.

Notation 2.9 Let $\Pi = \{\pi_0, \ldots, \pi_{i-1}, \pi_i, \pi_{i+1}, \ldots, \pi_n\}$, and $\Pi' = \{\pi_0, \ldots, \pi_{i-1}, \pi_i', \pi_{i+1}, \ldots, \pi_n\}$. We use the following expressions as shorthands:

- $\langle \Pi, \sigma \rangle \xrightarrow{\pi_i, \pi_i'} \langle \Pi', \sigma' \rangle$ for a transition $\langle \Pi, \sigma \rangle \longrightarrow \langle \Pi', \sigma' \rangle$ that is provable without using the transition rule for rule application,
- $\langle \Pi, \sigma \rangle \xrightarrow{\pi, \pi'}$ for the fact that there are Π', σ' such that $\langle \Pi, \sigma \rangle \xrightarrow{\pi, \pi'} \langle \Pi', \sigma' \rangle$;
  In case $\langle \Pi, \sigma \rangle \xrightarrow{\pi, \pi'}$, we say that π is *executable in* $\langle \Pi, \sigma \rangle$,
- $\langle \Pi, \sigma \rangle \xrightarrow{\pi_i, \gamma, \pi_i'} \langle \Pi', \sigma \rangle$ for a transition $\langle \Pi, \sigma \rangle \longrightarrow \langle \Pi', \sigma' \rangle$ that is provable by using the transition rule for rule application instantiated with γ, i.e. $\gamma = \pi \leftarrow \phi \mid \pi'$,
- $\langle \Pi, \sigma \rangle \xrightarrow{\pi, \gamma, \pi'}$ for the fact that there is a Π' such that $\langle \Pi, \sigma \rangle \xrightarrow{\pi, \gamma, \pi'} \langle \Pi', \sigma \rangle$; In case $\langle \Pi, \sigma \rangle \xrightarrow{\pi, \gamma, \pi'}$, we say that the rule-commitment pair $\langle \gamma, \pi \rangle$ is *applicable in* $\langle \Pi, \sigma \rangle$.

3 The AGENT-0 Interpreter

We define a language in which an interpreter for AGENT-0 can be programmed. This language is based on our paper [1]. It is an imperative language with operators for referring to the basic notions of the agent language itself and for programming selection strategies. Since no form of selection from rules or commitments is used in AGENT-0, we will not discuss selection actions, but refer the reader to [1].

3.1 Syntax

The commitment and rule terms of the meta language are used to access the commitments and rules of the agent program of the object language in the meta language. The terms refer to sets of commitments or sets of rules. The operators of the meta language for building complex terms are the usual set operators.

Definition 3.1 Let $\mathsf{Var}_\Pi, \mathsf{Var}_\Gamma$ be given sets of variables.
The set of *commitment terms* $\mathfrak{T}_\Pi$ is defined by: (i) $\mathsf{Var}_\Pi \subseteq \mathfrak{T}_\Pi$, (ii) $\emptyset, \Pi \in \mathfrak{T}_\Pi$, (iii) if $g_0, g_1 \in \mathfrak{T}_\Pi$, then $g_0 \cap g_1, g_0 \cup g_1, g_0 - g_1 \in \mathfrak{T}_\Pi$.
The set of *rule terms* $\mathfrak{T}_\Gamma$ is defined by: (i) $\mathsf{Var}_\Gamma \subseteq \mathfrak{T}_\Gamma$, (ii) $\emptyset, \Gamma \in \mathfrak{T}_\Gamma$, (iii) if $r_0, r_1 \in \mathfrak{T}_\Gamma$, then $r_0 \cap r_1, r_0 \cup r_1, r_0 - r_1 \in \mathfrak{T}_\Gamma$.

The commitment and rule terms are the usual set terms, constructed from the set operators $\cap$, etc. $\emptyset$ is a constant denoting the empty set. Furthermore, Γ is a rule term constant denoting the set of rules of an agent program. The commitment term Π, however, is a variable which denotes the commitments of the current object mental state during execution.

Definition 3.2 The set of *meta statements* $\mathfrak{S}$ is defined by:

- if $G \in \mathsf{Var}_\Pi, g \in \mathfrak{T}_\Pi$, then $G := g \in \mathfrak{S}$,
- if $g, g' \in \mathfrak{T}_\Pi$, then $g = g', g \neq g' \in \mathfrak{S}$,
- if $G, G' \in \mathsf{Var}_\Pi$, then $ex(G, G') \in \mathfrak{S}$,
- if $R \in \mathsf{Var}_\Gamma$ and $G, G' \in \mathsf{Var}_\Pi$, then $apply(R, G, G') \in \mathfrak{S}$
- if $R \in \mathsf{Var}_\Gamma, r \in \mathfrak{T}_\Gamma$, then $R := r \in \mathfrak{S}$,
- if $r, r' \in \mathfrak{T}_\Gamma$, then $r = r', r \neq r' \in \mathfrak{S}$,
- if $\beta, \beta' \in \mathfrak{S}$, then $\beta; \beta', \beta + \beta', \beta^* \in \mathfrak{S}$,

The meta language includes assignment of sets of commitments or rules to respectively commitment or rule variables, tests for equality on the commitment and rule terms, and the regular programming constructs for sequential composition, nondeterministic choice, and iteration. The meta language includes two actions *ex* and *apply* for respectively execution of committed actions and application of rules. These actions are calls to the object agent system to perform object transition steps corresponding to execution of actions or application of rules. The first argument position of the action *ex* should be filled with an input term denoting the set of committed actions from which to execute. The second argument position should be substituted with an output variable denoting a set of remaining commitments after (partly) execution. The first two argument positions of *apply* should be substituted with input terms denoting a set of rules and commitments respectively. The third argument position should be substituted with an output variable denoting the resulting set of commitments after application of the rules.

3.2 Semantics

In this section the operational semantics of the meta language is defined. The transition relation of the meta transition system is denoted by $\Longrightarrow$. The transition relation $\Longrightarrow$ is a relation on meta configurations, which are pairs consisting of a program statement and a meta state. Meta level states should include the information about object level features an agent interpreter should be able to access. Among these features are the object mental state and the commitment rules of an agent program. Furthermore, a meta state should keep track of the values of variables used in the meta program.

Definition 3.3 A *meta state*, or *m-state* τ is a tuple $\langle\langle \Pi, \sigma\rangle, <_\Pi, \Gamma, V\rangle$, where $\langle \Pi, \sigma\rangle$ is an *object mental state*, $<_\Pi$ is an ordering on the set of actions A, Γ is a set of *object rules*, and V is a *variable valuation* of type : $(\mathsf{Var}_\Pi \rightarrow \wp(\mathsf{A})) \cup (\mathsf{Var}_\Gamma \rightarrow \wp(\mathsf{R}))$.

The ordering on the set of actions is used to define priorities on action types, as will be explained below. A function *max* is used to select an action with highest priority and is defined by $max(X) = \{x \in X \mid$ there is *no* $x' \in X$ such that $x < x'\}$ where $<$ is an order on set X.

An *m-configuration* is a pair $\langle \beta, \tau\rangle$ where β is a program statement and τ is an m-state. We also write an m-configuration as a quadruple $\langle \beta, \langle \Pi, \sigma\rangle, <_\Pi, V\rangle$, where the constant set of rules is dropped from the configuration.

Definition 3.4 Let $\tau = \langle\langle \Pi, \sigma\rangle, <_\Pi, \Gamma, V\rangle$ be an m-state, and T range over goal and rule terms. Then the interpretation function $[\![\cdot]\!]_\tau : (\mathfrak{T}_\Pi \to \wp(\mathsf{A})) \cup (\mathfrak{T}_\Gamma \to \wp(\mathsf{R}))$ is defined by: (i) $[\![T]\!]_\tau = V(T) \cap \Pi$, for $T \in \mathsf{Var}_\Pi \cup \mathsf{Var}_\Gamma$, (ii) $[\![\Pi]\!]_\tau = \Pi$, $[\![\Gamma]\!]_\tau = \Gamma$, (iii) $[\![\emptyset]\!]_\tau = \emptyset$, and (iv) $[\![T_0 \oplus T_1]\!]_\tau = [\![T_0]\!]_\tau \oplus [\![T_1]\!]_\tau$, for $\oplus \in \{\cap, \cup, -\}$. We will drop the subscript referring to the state in the rest of this paper, as the context will make clear which state is referred to.

The semantics of the action $ex(G, G')$ is defined by iteration. $ex(G, G')$ is executed by choosing an executable action with highest priority from $[\![G]\!]$, deleting this action from $[\![G]\!]$, executing the action at object level, and returning the new action in output variable G', until no actions from $[\![G]\!]$ can be executed including the case $[\![G]\!] = \emptyset$.

Definition 3.5 *(execution rule for ex)*

$$\frac{\langle \Pi, \sigma\rangle \xrightarrow{\pi,\pi'} \langle \Pi', \sigma'\rangle, \pi \in max([\![G]\!])}{\langle ex(G, G'), \langle \Pi, \sigma\rangle, V\rangle \Longrightarrow \langle ex(G, G'), \langle \Pi', \sigma'\rangle, V\{([\![G]\!] \setminus \{\pi\})/G, ([\![G']\!] \cup \{\pi'\})/G'\}\rangle}$$

$$\frac{\langle \Pi, \sigma\rangle \not\xrightarrow{\pi,\pi'} \text{ for all } \pi \in [\![G]\!], \pi' \in \mathsf{A}}{\langle ex(G, G'), \langle \Pi, \sigma\rangle, V\rangle \Longrightarrow \langle E, \langle \Pi, \sigma\rangle, V\rangle}$$

The meta action $ex(G, G')$ is a call to the object agent language system to execute a maximal subset of actions from the set $[\![G]\!]$ and recording the result of executing those actions in G' (resulting in a change to the variable valuation V). The variable G is also changed and contains the remaining actions which are not executable in the current object state. The second transition rule specifies the termination condition of the iteration.

Much along the same lines as for the action *ex* we can define the semantics of the action *apply*. As before, an iterative definition of the semantics of *apply* is given. Informally, the action $apply(R, G, G')$ applies as much rules from $[\![R]\!]$ as possible, recording the change to the set of commitments as a result of this action in G'. We have to distinguish two cases. The first case concerns the application of a condition-action rule. In this case, a condition-action rule from $[\![R]\!]$ is applied, the rule is removed from $[\![R]\!]$, and the result is stored in the variable G'. The rule has to be removed from $[\![R]\!]$ to guarantee termination of *apply*. The second case concerns the case that a rule modifies a goal from G. In this case an arbitrary commitment from G is chosen and an arbitrary rule from R is chosen that is applicable to the chosen commitment; the commitment is deleted from $[\![G]\!]$, and the result of applying the rule is stored in the output variable G'. We require that for all variables G, $\Box \notin [\![G]\!]$.

Definition 3.6 *(execution rule for apply)*

$$\frac{\langle \Pi, \sigma\rangle \xrightarrow{\pi,\gamma,\pi'} \langle \Pi', \sigma\rangle, \pi = \Box}{\langle apply(R, G, G'), \langle \Pi, \sigma\rangle, V\rangle \Longrightarrow \langle apply(R, G, G'), \langle \Pi', \sigma\rangle, V\{([\![R]\!] \setminus \{\gamma\})/R, ([\![G']\!] \cup \{\pi'\})/G'\}\rangle}$$

$$\frac{\langle \Pi, \sigma\rangle \xrightarrow{\pi,\gamma,\pi'} \langle \Pi', \sigma\rangle, \pi \in [\![G]\!]}{\langle apply(R, G, G'), \langle \Pi, \sigma\rangle, V\rangle \Longrightarrow \langle apply(R, G, G'), \langle \Pi', \sigma\rangle, V\{([\![G]\!] \setminus \{\pi\})/G, ([\![G']\!] \cup \{\pi'\})/G'\}\rangle}$$

$$\frac{\langle \Pi, \sigma\rangle \not\xrightarrow{\pi,\gamma,\pi'} \text{ for all } \pi \in [\![G]\!], \gamma \in [\![R]\!], \pi' \in \mathsf{A}}{\langle apply(R, G, G'), \langle \Pi, \sigma\rangle, V\rangle \Longrightarrow \langle E, \langle \Pi, \sigma\rangle, V\rangle}$$

The third rule specifies the termination condition of the iteration.

For a definition of the semantics of the operators $+$, $*$, and $;$ we refer to [1].

3.3 Interpreter

The basic difference between a number of agent languages resides in their *control structure*. The main purpose of a control structure for agent languages is to specify which commitments to deal with first and which rules to apply during the execution of an agent program. The strategy for executing commitments and applying rules in AGENT-0, is to execute all executable commitments and apply all applicable rules in every cycle of the interpreter. In this section we use the meta language to define the AGENT-0 interpreter. The strategy used in AGENT-0 results in a simple meta program by using the actions *ex* and *apply*. The strategy corresponds to the PSR syllogism of section 2.2

The AGENT-0 interpreter continuously executes the two steps: (1) update the commitments (the rules of the agent program are used in this phase), and (2) execute the commitments (this phase is independent from the agent program). The update phase of the loop consists of two distinct steps. The commitments are updated by firing the applicable rules of the agent program, and the feasibility of the current commitments of the agent is checked. The feasibility of a commitment is a check on the mental state to see whether or not the agent believes it is capable of executing the commitment. The order imposed on these distinct steps in the AGENT-0 interpreter is not mentioned in [5]. However, the order proposed here, first firing applicable rules and then checking for feasibility, seems the most plausible thing to do.

The execution phase, i.e. the second phase in the loop above, boils down to executing as much commitments as possible in AGENT-0. Since the actions executed are simple actions we might presume that each of the actions executed is executed completely. This remark applies in particular to conditional actions, which are executed by first performing a test and in case the test succeeds executing the action part.

An implicit order on the type of actions is assumed. In particular, the refrain actions have a higher priority than private or conditional actions. The reason is that refrain actions should always be executed first to prevent actions from which the agent should refrain from being executed. The priority on actions is used in defining the semantics of the meta action *ex*.

	AGENT-0	
	REPEAT	
Step 1	$G := \Pi$; $R := \Gamma$;	Application Phase
	apply$(R, G, _)$;	- apply modifies Π!
Step 2	$G := \Pi$;	
	apply$(\Delta, G, _)$;	- feasibility check
Step 3	$G := \Pi$;	Execution Phase
	REPEAT	
	ex(G, G');	
	$G := G'$	
	UNTIL $G = \emptyset$;	
	UNTIL FALSE;	

Table 3.3

The meta program defining the interpreter is given in table 3.3. It implements the three separate steps of the interpreter. Step 1 corresponds to firing the set of rules Γ of the agent program. After initialising the variables G and R, this step is implemented by the meta action *apply* which fires as much rules as possible.

Step 2 corresponds to the check for feasibility of committed actions, and is again implemented by the meta action *apply*. A set of rules Δ is presupposed. Each of the rules in this set is of the form $a \leftarrow \neg\phi \mid$. The head of the rule corresponds to a type of action and the condition ϕ of the rule corresponds to the feasibility or capability conditions for that action. The body is empty, since if the capability conditions do not hold, i.e. $\neg\phi$ is satisfied, then the action should be removed from the set of commitments. By firing all applicable rules in Δ the check is performed on all commitments.

Step 3 corresponds to the execution of as much commitments as possible. The meta action *ex* implements this step of the interpreter. The order on actions is assumed to give higher priority to refrain actions than to any of the other action types. The actions are executed completely by iteratively repeating execution for the remaining part of (conditional) actions.

4 Conclusion

By abstracting from a number of features of AGENT-0, we have been able to construct an operational semantics for AGENT-0. We used a two-layered approach by separating the semantics of the basic programming constructs and the semantics of the control structure in the interpreter for AGENT-0. This approach yields a clear and intuitive definition for AGENT-0, as well as for other agent languages. This allows for a formal comparison, and thereby clarifies a number of differences between rule-based agent languages (cf. [1]).

The specification of a formal semantics for AGENT-0 also resulted in a better understanding of the use of rules in AGENT-0. We distinguished a bottom-up or *rule-driven* approach used in AGENT-0 and a top-down or *goal-driven* approach used in AgentSpeak(L) and 3APL.

References

[1] Koen V. Hindriks, Frank S. de Boer, Wiebe van der Hoek, and John-Jules Ch. Meyer. Control Structures of Rule-Based Agent Languages. *Accepted for ATAL98*, 1998.

[2] Koen V. Hindriks, Frank S. de Boer, Wiebe van der Hoek, and John-Jules Ch. Meyer. Formal Semantics for an Abstract Agent Programming Language. In *Intelligent Agents IV (LNAI 1365)*, pages 215–229, 1998.

[3] G. Plotkin. A structural approach to operational semantics. Technical report, Aarhus University, Computer Science Department, 1981.

[4] Anand S. Rao. AgentSpeak(L): BDI Agents Speak Out in a Logical Computable Language. In *Agents Breaking Away*, pages 42–55. Springer, 1996.

[5] Yoav Shoham. Agent-oriented programming. *Artificial Intelligence*, 60:51–92, 1993.

[6] Sarah Rebecca Thomas. *PLACA, An Agent Oriented Programming Language*. PhD thesis, Department of Computer Science, Stanford University, 1993.

Model refinement and model checking for $S5^n$

Alessio Lomuscio and Mark Ryan
School of Computer Science
University of Birmingham
Birmingham B15 2TT, UK
www.cs.bham.ac.uk/{~arl,~mdr}

15 June 1998

Abstract

Halpern and Vardi have proposed the notion of refinement of $S5^n$ Kripke models in order to solve multi-agent problems in which knowledge evolves. We argue that there are some problems with their proposal and attempt to solve them by moving from Kripke models to their corresponding *trees*. We define refinement of a tree with a formula, show some properties of the notion, and illustrate with the muddy children puzzle.

1 Introduction

The modal logic $S5^n$ (see for example [HC96]), also known as $KT45^n$, has been used to model knowledge in multi-agent systems (MAS) for some years now [Hin62, FHMV95]. $S5^n$ is a classical modal logic containing n modalities expressing private knowledge (written $\Box_i$, $1 \leqslant i \leqslant n$), and operators for expressing common knowledge and distributed knowledge within a group.

The standard (*consequence relation*) approach to using $S5^n$ is to set up a situation as a set of formulas Γ, and to attempt to show that the situation satisfies a property ϕ by establishing $\Gamma \vdash \phi$ or $\Gamma \models \phi$. Establishing $\Gamma \vdash \phi$ involves finding a proof of ϕ from Γ, while establishing $\Gamma \models \phi$ involves reasoning about all (usually infinitely many) Kripke models satisfying Γ to show that they also satisfy ϕ. The completeness of $S5^n$ shows that these two notions are equivalent. However, experience has shown that this approach is computationally very expensive.

In order to overcome the intractability of this approach, Halpern and Vardi have proposed to use *model checking* as an alternative to theorem proving [HV91]. In the model checking approach, the situation to be modelled is codified as a single Kripke model M rather than as a set of formulas Γ. The task of verifying that a property ϕ holds boils down to checking that M satisfies ϕ, written $M \models \phi$. This task is computationally much easier than the theorem proving task, being linear in the size of M and the size of ϕ [HV91].

Halpern and Vardi informally illustrate their approach by modelling the muddy children puzzle. In that puzzle, there are n children and n atomic propositions $p_1, p_2, \ldots, p_n$ representing whether each of the children have mud on their faces or not. Various announcements are made, first by the father of the children and then by the children themselves. The children thus acquire information about what other children know, and after some time the muddy ones among them are able to conclude that they are indeed muddy. We describe the problem in greater detail below.

Halpern and Vardi propose the following way of arriving at the model M to be checked. They start with the most general model for the set of atomic propositions at hand. In order to deal with the announcements made, they successively *refine* the model with formulas expressing the announcements made. This refinement process consists of removing some links from the Kripke model. At any time during this process, they can check whether child i knows p_i (for example), by checking whether the current model satisfies $\Box_i p_i$.

This method is illustrated in the paper [HV91] and the book [FHMV95], but a precise definition of the refinement operation is not given. Our original aim for this paper was to provide such a definition and explore its properties. However, we soon came to the opinion that there is no definition of model refinement on arbitrary $S5^n$ Kripke structures that will have intuitively acceptable properties. We explain our reasons for this view in section 2. We believe the refinement and model checking ideas can still be made to work, however. In section 3 we introduce a structure derived from a Kripke model, which we call a *Kripke tree*, and define the refinement operation on Kripke trees. We illustrate this notion using the muddy children example in section 4. In section 5 we state and prove some properties of the refinement operation on Kripke trees, and conclude in section 6.

1.1 Syntax and semantics

We assume finite sets P of *propositional atoms*, and A of *agents*. Formulas are given by the usual grammar:

$$\phi ::= p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \Box_i\phi \mid C\phi$$

where $p \in P$ and $i \in A$. Intuitively the formula $\Box_i\phi$ represents the situation in which the agent i knows the fact represented by the formula ϕ. The other propositional connectives can be defined in the usual way. As The modal connectives E and B are defined as:

$$\begin{array}{rcl} E\phi & \text{means} & \bigwedge_{i\in A} \Box_i\phi \\ \Diamond_i\phi & \text{means} & \neg\Box_i\neg\phi \\ B\phi & \text{means} & \neg C\neg\phi \end{array}$$

$E\phi$ means that everyone knows ϕ, $\Diamond_1\phi$ means "it is consistent with 1's knowledge that ϕ", while $C\phi$ is the much stronger statement that ϕ is common knowledge. In a multi-agent setting, a formula ϕ is said to be common knowledge if it is known by all the agents, and moreover that each agent knows that it is known by all the agents; and moreover, each agent knows that fact, and that one, etc. A general announcement of ϕ results in common knowledge of ϕ among the hearers, because as well as hearing ϕ they also see that the others have heard it too (we assume throughout that all the agents are perceptive, intelligent, truthful). If one agent secretly informs all the others of ϕ, the result will be that everyone knows ϕ, but ϕ will not be common knowledge. B is the dual of C. Although not particularly useful intuitively, we will need it for technical reasons.

Definition 1.1 A formula is *universal* if it has only the modalities $C, E, \Box_i$ and no negations outside them.

Definition 1.2 An $S5^n$ *Kripke model* $M = (W, \sim, \pi, w)$ of the modal language over atomic propositions P and agents A is given by:

1. A set W, whose elements are called *worlds*;

2. An A-indexed family of relations $\sim = \{\sim_i\}_{i \in A}$. For each $1 \leqslant i \leqslant n$, $\sim_i$ is an equivalence relation on W ($\sim_i \subseteq W \times W$), called the *accessibility relation*;

3. A function $\pi : W \to \mathcal{P}(P)$, called the *assignment function*;

4. A world $w \in W$, the *actual world.*

See figure 1 for an illustration.

Let $x \in W$. We define the relation of satisfaction of ϕ by M at x, written $M \models_x \phi$, in the usual way:

$$\begin{array}{rl} M \models_x p & \text{iff } p \in \pi(x) \\ M \models_x \neg\phi & \text{iff } M \not\models_x \phi \\ M \models_x \phi \wedge \psi & \text{iff } M \models_x \phi \text{ and } M \models_x \psi \\ M \models_x \Box_i \psi & \text{iff for each } y \in W,\ x \sim_i y \text{ implies } M \models_y \psi \\ M \models_x C\psi & \text{iff for each } k \geqslant 0 \text{ and } i_1, i_2, \ldots, i_k \in A, \text{ we have } M \models_x \Box_{i_1} \ldots \Box i_k \psi \end{array}$$

We say that y is *reachable in k steps* from x if there are $w_1, w_2, \ldots w_{k-1} \in W$ and $i_1, i_2, \ldots i_k$ in A such that $x \sim_{i_1} w_1 \sim_{i_2} w_2 \cdots \sim_{i_{k-1}} w_{k-1} \sim_{i_k} y$. We also say that y is *reachable* from x if there is some k such that it is reachable in k steps. The following fact is useful for understanding the technical difference between E and C.

Theorem 1.3 ([FHMV95])

1. $M \models_x E^k\phi$ iff for all y that are reachable from x in k steps, we have $M \models_y \phi$.

2. $M \models_x C\phi$ iff for all y that are reachable from x, we have $M \models_y \phi$.

2 Refining Kripke models

Halpern and Vardi propose to refine Kripke models in order to model the evolution of knowledge. They illustrate their method with the muddy children puzzle.

2.1 The muddy children puzzle

There is a large group of children playing in the garden. A certain number (say k) get mud on their foreheads. Each child can see the mud on others (if present) but not on his own forehead. If $k > 1$ then each child can see another with mud on its forehead, so each one knows that at least one in the group is muddy. The father first announces that at least one of them is muddy [which, if $k > 1$, is something they know already]; and then he repeatedly asks them 'Does any of you know whether you have mud on your own forehead?' The first time they all answer 'no'. Indeed, they go on answering 'no' to the first $k - 1$ questions; but at the kth those with muddy foreheads are able to answer 'yes'.

At first sight, it seems rather puzzling that the children are eventually able to answer the father's question positively. The clue to understanding what goes on lies in the notion of common knowledge. Although everyone knows the content of the father's initial announcement, the father's saying it makes it common knowledge among them, so now they all know that everyone else knows it, etc. Consider a few cases of k.

$k = 1$, i.e. just one child has mud. That child is immediately able to answer 'yes', since she has heard the father and doesn't see any other child with mud.

$k = 2$, say a and b have mud. Everyone answers 'no' the first time. Now a thinks: since b answered 'no' the first time, he must see someone with mud. Well, the only person I can see with mud is b, so if b can see someone else it must be me. So a answers 'yes' the second time. b reasons symmetrically about a, and also answers 'yes'.

$k = 3$, say a, b, c. Everyone answers 'no' the first two times. But now a thinks: if it was just b and c with mud, they would have answered 'yes' the second time. So there must be a third person with mud; since I can only see b, c having mud, the third person must be me. So a answers 'yes' the third time. For symmetrical reasons, so do b, c.

And similarly for other cases of k.

To see that it was not common knowledge before the father's announcement that one of the children was muddy, consider again $k = 2$, say a, b. Of course a and b both know someone is muddy (they see each other), but, for example, a doesn't know that b knows that someone is dirty. For all a knows, b might be the only dirty one, and therefore not be able to see a dirty child.

2.1.1 The formalisation in [HV91]

Suppose $A = \{1, \ldots n\}$ and $P = \{p_1, \ldots, p_n\}$; p_i means that the ith child has mud on its forehead. Suppose $n = 3$. The assumption of this puzzle is that each child can see the other children but cannot see itself, so each child knows whether the others have mud or not, but does not know about itself. Under these assumptions, Halpern and Vardi propose the Kripke structure of figure 1 to model the initial situation.

Let w be any world in which there are at least two muddy children (i.e. w is one of the four upper worlds). In w, every child knows that at least one of the children has mud. However, it is not the case that it is common knowledge that each child has mud, since the world at the bottom of the lattice is reachable (cf. theorem 1.3).

To model the father's announcement, Halpern and Vardi refine the model M_1 in figure 1, arriving at M_2 in figure 2. The refinement process is not precisely defined in [HV91, FHMV95], though arguments in favour of the transformation from M_1 to M_2 are given (these figures also appear in [HV91, FHMV95]).

Suppose now that the father asks the children whether they know whether they are muddy or not, and the children answer simultaneously that that they do not. Halpern and Vardi argue that this renders all models in which there is only one muddy child inaccessible, resulting in M_3 (figure 3).

If there are precisely two children with mud (i.e. the actual world is one of the three in the second layer), then each of the muddy children now knows it is muddy. For suppose the actual world is the left one of those three, i.e. w with $\pi(w) = \{p_1, p_2\}$. We easily verify that $M_3 \models_w \Box_1 p_1$ and $M_3 \models_w \Box_2 p_2$.

If all three children are muddy, i.e. the actual world w is the top one, then we are not yet done, for we do not have $M_3 \models_w \Box_i p_i$ for any i. The father again asks each of the children if they know if they are muddy, and the model is refined again according to their answer "no", resulting in M_4 which is M_3 with the last remaining links removed. We can easily check that $M_4 \models_w \Box_i p_i$ for each i.

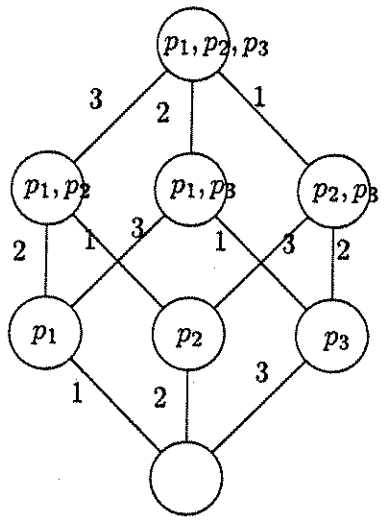


Figure 1: M_1: The Kripke model for the muddy children puzzle with $n = 3$.

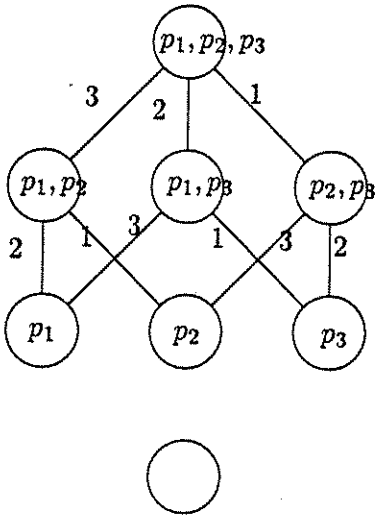


Figure 2: M_2: The Kripke structure after the father speaks.

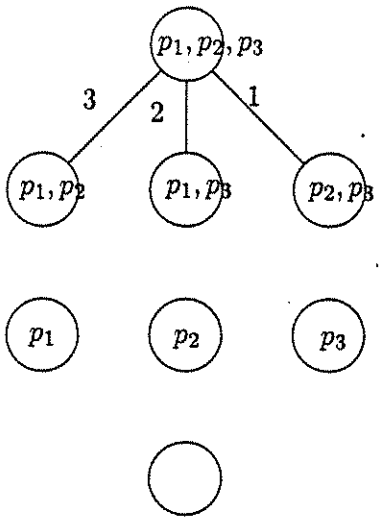


Figure 3: M_3: The Kripke structure after the children announce that they don't know whether they are muddy.

In summary, the method proposed by Halpern and Vardi for solving muddy-children-type puzzles is the following. Start with a suitably general model M_1 reflecting the initial set-up of the puzzle. Refine it successively by the announcements made. At the end of the announcements, check formulas against the refined model. In the example above, we refined M_1 first by $\phi_1 = C(p_1 \vee p_2 \vee p_3)$ (the father's announcement), and then twice by

$$\phi_2 = C(\neg\Box_1 p_1 \wedge \neg\Box_1 \neg p_1) \wedge C(\neg\Box_2 p_2 \wedge \neg\Box_2 \neg p_2) \wedge C(\neg\Box_3 p_3 \wedge \neg\Box_3 \neg p_3)$$

which corresponds to each of the three children announcing that they don't know whether they are muddy or not.

Halpern and Vardi do not precisely define what refinement by a formula means. The intuition they give is that refinement removes a minimal set of links of the model, so that the model satisfies the formula at the actual world. Removing links means that epistemic possibilities are removed, that is, knowledge is gained, so this seems intuitively the right thing to do.

2.2 Problems with refinement of Kripke models

Let us write $M * \phi$ to denote the result of refining the model M by the formula ϕ. Thus, in the example above, $M_2 = M_1 * \phi_1$, etc.

Our original aim was to make precise this notion of refinement of a Kripke model by a formula, and to investigate its properties. We investigated several possible definitions: essentially a refinement procedure will remove the links to the states that are responsible for the non-satisfaction of the formula we are refining with. However, we quickly came upon examples which showed that it will not be easy to achieve all the reasonable properties one could wish for.

Example 2.1 Let M_5 be the Kripke model illustrated in figure 4, with the left-hand world w the actual world, and consider refining by $\Box_1 p$. The definitions we examined differed in

Figure 4: M_5 and M_6 (example 2.1)

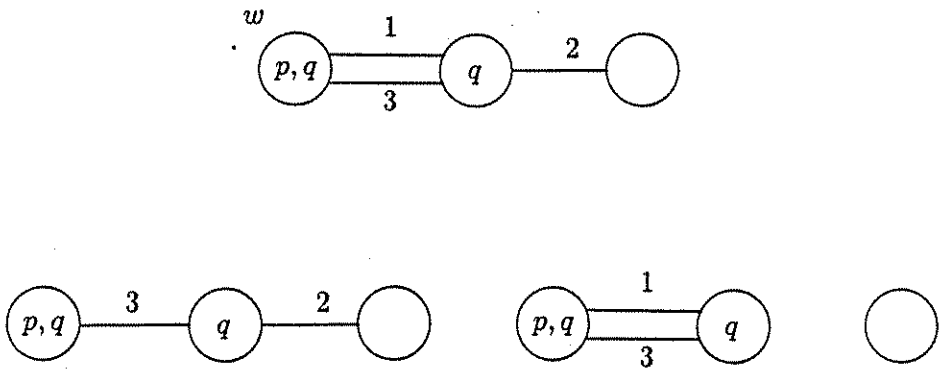


Figure 5: Two outcomes for refinement of the top model by $\Box_1\Box_2(p \vee q)$ (example 2.2)

subtle cases involving quite complex formulas and models, but they all agreed in this one: the resulting model must be M_6 (see figure). What happens is that agent 1 gains the knowledge of p, and so must eliminate the epistemic possibility of $\neg p$ by removing the link.

The counterintuitive property of this example is that $M_5 \models_w \Box_3\Diamond_1 p$, while $M_6 \not\models_w \Box_3\Diamond_1 p$. Thus, in M_5, agent 3 knows that p is consistent with 1's knowledge. But after 1 learns p for sure in M_6, 3 no longer knows this!

Example 2.2 Figure 5 shows a model and (the only) two outcomes one could consider for its refinement by $\Box_1\Box_2(p \vee q)$. One must remove either the 1 link or the 2 link in order to prevent the 1–2 path to the world exhibiting $\neg(p \vee q)$. The choice is which link to remove. Both outcomes reveal undesirable properties of the refinement operator. In the first case, removing the 1 link adds too much to 1's knowledge (he is now able to rule out everything but the current world), while the second case gives us a situation in which a model satisfies $\Box_3\Diamond_2\neg q$ but its refinement by $\Box_1\Box_2(p \vee q)$ does not. It is counterintuitive that 3's knowledge should change in this way when we refine by $\Box_1\Box_2(p \vee q)$.

The second case at least has the desirable property that a minimal change of the knowledge of agents at the actual world w is made, since the set of reachable states from w is maximised (cf. theorem 1.3).

Example 2.3 Refinement by universal formulas ought to be cumulative, and such formulas ought to commute with each other (i.e. $M * \phi * \psi = M * \psi * \phi$). However, another example shows that this will be hard to achieve. Consider the model M_7 shown at the top of figure 6, and let $\phi = \Box_1 p$ and $\psi = \Box_1\Box_2(p \vee q)$. Whatever way one thinks about defining $*$, the result in the left-hand branch seems clear. Note that $M_7 * \Box_1 p$ already satisfies $\Box_1\Box_2(p \vee q)$ and therefore $M_7 * \Box_1 p * \Box_1\Box_2(p \vee q) = M_7 * \Box_1 p$.

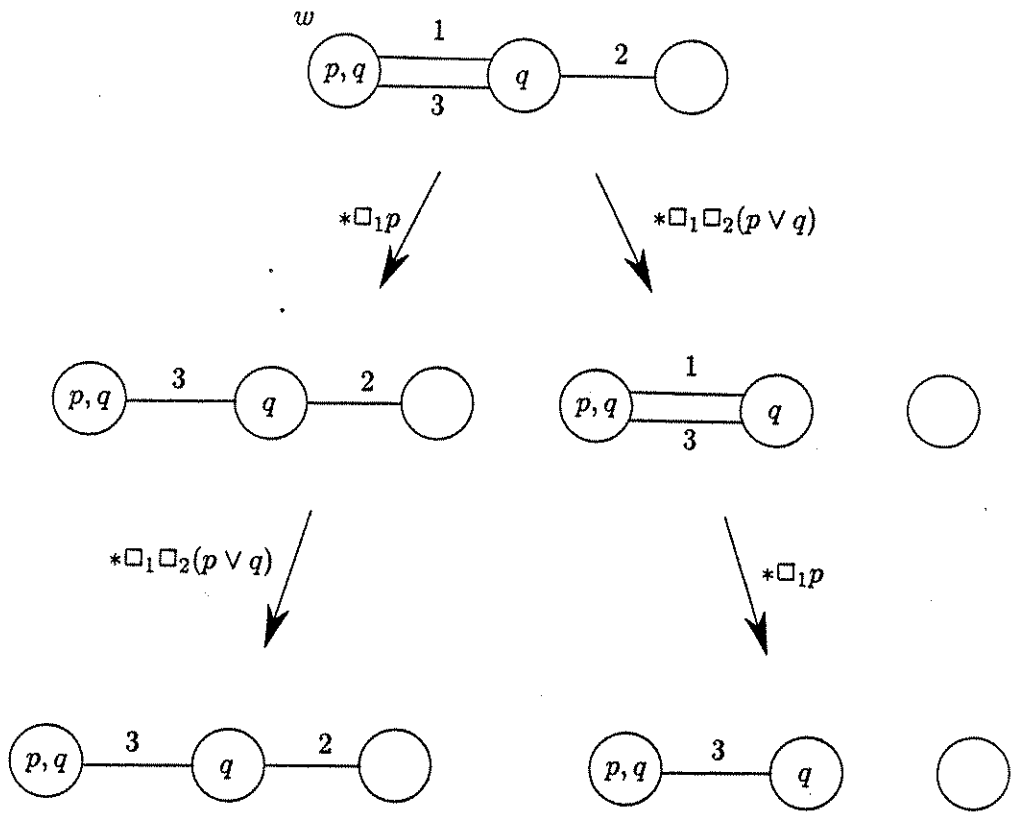


Figure 6: Two evolutions of M_7 (example 2.3), showing that $M * \phi * \psi \neq M * \psi * \phi$

An argument for the stated result of $M_7 * \Box_1\Box_2(p \vee q)$ was given in example 2.2, and further refining by $\Box_1 p$ leaves little room for maneuver. The resulting models differ on whether they satisfy (for example) $\Box_3\Box_2 q$.

Example 2.3 shows that even universal formulas (definition 1.1), do not enjoy commutativity in any reasonable refinement setting. However, commutativity for universal formulas seems intuitively correct: the order in which ideal agents acquire information should not matter. Non-universal formulas are a different matter, since they can express absence of knowledge, and this will not commute with the acquisition of new knowledge.

3 Refining Kripke trees

Some of the problems exhibited by the three examples at the end of the preceding section seem to be due to the following fact: when we remove a link in a Kripke model in order to block a certain path, we also block other paths that used that link. To overcome this problem, we would like to unravel Kripke models into trees, in which each link participates in just one path. At first sight this looks like it will destroy the finiteness of our models, a feature on which effective refinement operators and model checking operators rely. To retain finiteness, we will need to limit in advance the maximum nesting of boxes that is allowed, and construct a tree to depth greater than this number. Semantic structures similar to Kripke trees have

been defined in [HC84]. Our definition differs in detail from the one in [HC84], but it largely agrees with it in spirit.

In this section we define the notion of Kripke tree, show a translation of equivalence Kripke models into Kripke trees, define and algorithm for refining knowledge structures and prove a few properties about it.

3.1 Kripke trees: basic definitions

Definition 3.1 (Kripke Tree) Let $M = (W, \sim, \pi, w_0)$ be an S5^n Kripke model. The *Kripke tree* $T_M = (V, E, \sigma)$ generated by M is given as follows:

- The set of *vertices* is the set of paths in M:

$$V = \left\{ (w_0, i_1, w_1, \ldots, w_{k-1}, i_k, w_k) \;\middle|\; \forall j.\, w_j \in W,\, i_j \in A,\, w_j \sim_{j+1} w_{j+1} \right\}$$

- E is an A-indexed family of sets of edges. For $s, s' \in V$, there is an i-edge between s, s', written $(s, s') \in E_i$, iff s' equals s extended by an i link, i.e. $s = (w_0, i_1, w_1, \ldots, w_k), s' = (w_0, i_1, \ldots, w_k, i, w)$ for some w.

- The *valuation* σ is defined by $\sigma((w_0, i_1, w_1, \ldots, w_k)) = \pi(w_k)$.

The vertex $w_0 \in V$ is called *root* of the tree. We also allow the empty tree $(\varnothing, \varnothing, \varnothing)$ which we write as $\perp$. It has no root. When the model M is clear from the context or not relevant we will simply indicate the tree as T.

The sets E_i can be omitted from definition 3.1, since they can be derived from V. We retain them for convenience.

Kripke trees are irreflexive, intransitive, anti-symmetric, anti-convergent and serial.

If M has at least two distinct worlds related by some $\sim_i$, then T_M is infinite. For our purposes of model refinement, we usually want to deal with finite trees. T_M^k is T_M with paths truncated at length k. Obviously by truncating the tree we will lose seriality.

Definition 3.2 (Truncated tree of depth k) Given an tree $T_M = (V, E, \sigma)$, the truncated tree of depth k is defined as $T_M^k = (V', E', \sigma')$, where

- $V' = \{(w_0, i_1, w_1, \ldots, w_{j-1}, i_j, w_j) \in V \mid j \leqslant k\}$.
- $E' = E|_{V'}$ is the restriction of E to V',
- $\sigma' = \sigma|_{V'}$ is the restriction of σ to V'.

Infinite and finite trees satisfy modal formulas in the expected way:

Definition 3.3 (Interpretation) Let ϕ be an S5^n formula. The satisfaction of ϕ by T at vertex v, written $T \models_v \phi$, is inductively defined as follows:

- $T \models_v p$ if $p \in \sigma(v)$;
- $T \models_v \neg\phi$ if not $T \models_v \phi$;
- $T \models_v \phi \wedge \psi$ if $T \models_v \phi$ and $T \models_v \psi$;

- $T \models_v \Box_i \phi$ if $\forall v' \in V$, $(v, v') \in E_i$ implies $T \models_{v'} \phi$;
- $T \models_v C\phi$ if $\forall v' \in V\, T_{v'} \models \phi$.

The tree T satisfies ϕ, written $T \models \phi$, if it satisfies ϕ at its root. The empty tree $\perp$ satisfies no formula.

An infinite tree T_M is semantically equivalent to its generating model M as the following shows:

Lemma 3.4 Let $M = (W, \sim, \pi, w_0)$ be an equivalence Kripke model and $T_M = (V, E, \sigma)$ its associated Kripke tree. Let $v = (w_0, i_1, w_1, \ldots, w)$ be any vertex ending in w, and ϕ any formula. Then:

$$M \models_w \phi \quad \text{iff} \quad T_M \models_v \phi.$$

Proof There is a one-to-one corespondence between paths in M from w and paths in T from v. □

Corollary 3.5 $M \models \phi$ if and only if $T_M \models \phi$.

For the case of truncated tree, Lemma 3.4 is not valid. However, we can prove a related result for formulas up to a certain level of modal nesting.

We inductively define the *rank* of a formula as follows:

Definition 3.6 (Rank of a formula) The rank $\text{rank}(\phi)$ of a formula ϕ is defined as follows:

- $\text{rank}(p) = 0$, where p is a propositional atom.
- $\text{rank}(\neg\phi) = \text{rank}(\phi)$.
- $\text{rank}(\phi_1 \wedge \phi_2) = max\{\text{rank}(\phi_1), \text{rank}(\phi_2)\}$.
- $\text{rank}(\phi_1 \vee \phi_2) = max\{\text{rank}(\phi_1), \text{rank}(\phi_2)\}$.
- $\text{rank}(\Box_i\phi) = \text{rank}(\phi) + 1$.
- $\text{rank}(C\phi) = \infty$.

The rank of a formula ϕ intuitively represents the maximum number of nested modalities that occur in ϕ. If an operator C occurs in ϕ we take the value of $\text{rank}(\phi)$ to be infinite. The rank of a formula reflects the maximal length of any path that needs to be explored to evaluate ϕ on an infinite tree. In other words to evaluate a formula ϕ of rank k at w_0 we need not examine worlds whose distance from w_0 is greater than k, where distance here is the number of points which appear in the minimal path connecting the two points. The following lemma formalises this.

Lemma 3.7 If $\text{rank}(\phi) \leqslant k$, $M \models \phi$ if and only if $T_M^k \models \phi$.

Proof By corollary 3.5, $M \models \phi$ if and only if $T_M \models \phi$, but, by induction, the evaluation of a formula of $\text{rank}(\phi) \leqslant k$ does not involve the evaluation of nodes of depth greater than k. So $T_M \models \phi$ if and only if $T_M^k \models \phi$, which gives the result. □

In the following we shift our attention from an equivalence Kripke model to its generated tree. It is possible to do so, because generated trees satisfy $S5^n$-axioms as the following shows:

Lemma 3.8 Let M be an equivalence model and T_M^k its generated model truncated at k.

1. $T_M^k \models \phi$, where ϕ is a tautology, and $\text{rank}(\phi) \leqslant k$.
2. $T_M^k \models \Box_i(\phi \Rightarrow \psi) \Rightarrow \Box_i\phi \Rightarrow \Box_i\psi$, and $\max\{\text{rank}(\phi), \text{rank}(\psi)\} \leqslant k-1$.
3. $T_M^k \models \Box_i\phi \Rightarrow \phi$, and $\text{rank}(\phi) \leqslant k-1$.
4. $T_M^k \models \Box_i\phi \Rightarrow \Box_i\Box_i\phi$, and $\text{rank}(\phi) \leqslant k-2$.
5. $T_M^k \models \Diamond_i\phi \Rightarrow \Box_i\Diamond_i\phi$, and $\text{rank}(\phi) \leqslant k-2$.
6. If for every vertex $v \in V$ of T_M^k, $T_M^k \models_v \phi$, then $T_M^k \models_v \Box_i\phi$, for any $i \in A$.
7. If $T_M^k \models \phi$, and $T_M^k \models \phi \Rightarrow \psi$ then $T_M^k \models \psi$.

Proof We prove item number 5. The others can be done similarly. Suppose the formula is not true on T^k with root v, and suppose $T_v^k \models \Diamond_i\phi$, but $T_v^k \not\models \Box_i\Diamond_i\phi$. So there is a point $v' \in V$ such that $T_{v'}^k \models \phi$, and $(v,v') \in E_i$. But $T_v^k \models \neg\Box_i\Diamond_i\phi$, so $T_v^k \models \Diamond_i\Box_i\neg\phi$, so there exists a $v'' \in V$ such that $T_{v''}^k \models \Box_i\neg\phi$, and $(v,v'') \in E_i$. But then since T^k originates from M, by lemma this must contain three points w, w', w'', corresponding to v, v', v'', such that $M \models_{w'} \phi$, and $M \models_{w''} \Box_i\neg\phi$ (theorem 3.4). But $w' \sim_i w''$ since $\sim_i$ is an equivalence relation. This is absurd. □

So Kripke trees are models for $S5_n$.

Before we proceed further, we introduce a few basic definitions and operations on subtrees.

Definition 3.9 (Rooted-subtrees) Let $T' = (V', E', \sigma')$, $T = (V, E, \sigma)$ be trees. T' is a *rooted subtree* of T, written $T' \leqslant T$, if $V' \subseteq V$, and $E|_{V'} = E'$, and $\sigma|_{V'} = \sigma'$.

Definition 3.10 (Intersection of trees) Let $T' = (V', E', \sigma')$ and $T = (V, E, \sigma)$ be trees such that $\sigma|_{V' \cap V} = \sigma'|_{V' \cap V}$. The intersection of T and T' is $T \sqcap T' = (V' \cap V, E' \cap E, \sigma'|_{V' \cap V})$.

It is easy to see that definition 3.10 (when applicable) defines a tree.

Definition 3.11 (Restriction of trees) Let $T = (V, E, \sigma)$ be a tree with root v, and V' a subset of V. The restriction of T to V', written $T|_{V'}$, is the largest subtree of T generated by v whose vertices are in V'..

The term "generated" in definition 3.11 is intended in the usual sense defined in the literature (see for example [Gol87] page 10). If the root of T is not in V', then $T|_{V'} = \bot$.

3.2 Kripke trees: refinement

In section 2.2, we discussed the difficulties that arise when using $S5_n$ Kripke models as knowledge structures for refinement. Example 2.3 showed that any straightforward procedure to refine an equivalence Kripke model will be non-commutative, i.e. there will be α, β, such that $M * \alpha * \beta \not\equiv M * \beta * \alpha$.

Commutativity can be achieved by shifting to Kripke trees. Before we can show this, we must define refinement on Kripke trees.

The typical working scenario in which we operate is the same one as that advocated by [HV91], except that we refine T_M instead of M. It can be described as follows: we are given an initial configuration of a MAS, and a set of formulas $\{\phi_1, \ldots, \phi_m\}$ that represent the update of the scenario. The question is whether the updated configuration will validate a set of formulas $\{\psi_1, \ldots, \psi_l\}$. We assume every ψ to have finite rank, i.e. we cannot check a formula containing the operator of common knowledge. There is no restriction on the ϕs.

Our method operates as follows:

1. Start from the most general equivalence Kripke model M that represents the MAS.
2. Generate the infinite tree T_M, as given in Definition 3.1.
3. Generate from T_M^k, the truncated tree of depth k, for some sufficiently large k.
4. Sequentially refine T_M^k with $\{\phi_1, \ldots, \phi_m\}$,
5. Check whether the resulting tree structure satisfies $\{\psi_1, \ldots, \psi_l\}$.

The method describes above needs some further explanation. First, what is the most general Kripke model representing a MAS configuration? How are we to build it? Our answer is the same as that given by Halpern and Vardi. Assume the set of atoms P is finite. We take the model whose universe W is equal to 2^P, i.e. the universe will cover all the possible assignments to the atoms. We take $\sim_i, i \in A$ to be the universal relations on $W \times W$, and w_0 to be the actual world of the given MAS.

In general we will require that M is more specific than the most general model, e.g. some agent will have a certain knowledge about the world. We can add all the formulas that need be satisfied to the set of updates $\{\phi_1, \ldots, \phi_m\}$. For example in the muddy children example we can start from the model with universal relations and add

$$\bigwedge_{i=1}^{3} C(p_i \Rightarrow K_i p_i)$$

to the set of updates.

We have already explained how to execute steps 1, 2, 3, and 5. We now present a notion of refinement to execute step 4.

Definition 3.12 (Refinement of Kripke tree structures) Given a truncated Kripke tree $T^m = (V, E, \sigma)$, a point $v \in V$, and a formula ϕ, the result $T' = (T, v) * \phi$ of refining T by ϕ at v is procedurally defined as follows. We assume that the negation symbols in ϕ apply only to atomic propositions (to achieve this, negations may be pushed inwards using de Morgan laws and dualities $\Box/\Diamond$ and C/B).

- If $T = \bot$, then $T' = \bot$.

- If $T \models_v \phi$, then $T' = T$.
- Otherwise the result is defined inductively on ϕ:
  - $\phi = p$. If $p \in \sigma(v)$, then $T' = T$, else $T' = \bot$.
  - $\phi = \neg p$. If $p \notin \sigma(v)$ then $T' = T$, else $T' = \bot$.
  - $\phi = \psi \wedge \chi$. $T' = ((T,v) * \psi) \sqcap ((T,v) * \chi)$.
  - $\phi = \psi \vee \chi$. If $(T,v) * \psi \leqslant (T,v) * \chi$ then $T' = (T,v) * \chi$, and if $(T,v) * \chi \leqslant (T,v) * \psi$ then $T' = (T,v) * \psi$. Otherwise T' is non-deterministically given as $(T,v) * \psi$ or $(T,v) * \chi$.
  - $\phi = \Box_i \psi$. T' is given by computing as follows:

$$
\begin{array}{l}
T' := T; \\
\text{for each } v' \text{ such that } (v,v') \in E_i \text{ do} \\
\quad \text{if } (T',v') * \psi = \bot, \text{ then} \\
\quad\quad T' := T'|_{V-\{v'\}} \\
\quad \text{else} \\
\quad\quad T' := (T',v') * \psi
\end{array}
$$

  - $\phi = \Diamond_i \psi$. Let X be the set $X = \{(T,v') * \phi \mid (v,v' \in E_i)\}$.

    If $X = \varnothing$, then $T' = \bot$,
    else T' is nondeterministically chosen to be a $\leqslant$-maximal element of X.
  - $\phi = C\psi$. T' is given by computing as follows:

$$
\begin{array}{l}
T' := T; \\
\text{for each } v' \in V \text{ do} \\
\quad \text{if } (T',v') * \psi = \bot, \text{ then} \\
\quad\quad T' := T'|_{V-\{v'\}} \\
\quad \text{else} \\
\quad\quad T' := (T',v') * \psi
\end{array}
$$

  - $\phi = B\psi$. Let X be the set $X = \{(T,v') * \phi \mid v' \in V\}$.

    If $X = \varnothing$, then $T' = \bot$,
    else T' is nondeterministically chosen to be a $\leqslant$-maximal element of X.

$T * \phi$ means $(T,v) * \phi$, where v is the root of T.

Lemma 3.13 Given a tree T, a formula α and a point v, $(T,v) * \alpha$ is a tree.

Proof Follows from the fact that if T is a tree then $T|_{V'}$ is also a tree. □

The intuition behind $(T,v) * \phi$ is that it is obtained by removing as small a set of links from T as possible, in order to satisfy ϕ. Note that, due to the clauses for $\vee, \Diamond_i, B$, $(T,v) * \phi$ is not uniquely defined. However, we will see that running the procedure on the muddy children example does not introduce nondeterminism.

4 The muddy children puzzle using Kripke trees

In section 2.1, we described the muddy children puzzle and we reported the formalisation that was given in [FHMV95, HV91]. Aim of the present section is to solve an instance of it (where the actual situation is coded by the tuple p_1, p_2, p_3 that we equivalently write as $(1,1,1)$ - all the children are muddy) by using Kripke trees and the methods we introduced in section 3.1. We follow the method that we defined in Section 3.2.

We start with the most general model to represent the puzzle: this is the model M_1 of figure 1[1]. Given M_1, we generate the infinite tree T_{M_1} for M_1 and then the truncation T_1 of M_1. In this example, we only need three levels to be unravelled. The starting tree and the two successive refinements are in figure 7. Let $\phi_1 = C(p_1 \vee p_2 \vee p_3)$ (this is the father's announcement), and $\phi_2 = C(\neg\Box_1 p_1 \wedge \neg\Box_1\neg p_1) \wedge C(\neg\Box_2 p_2 \wedge \neg\Box_2\neg p_2) \wedge C(\neg\Box_3 p_3 \wedge \neg\Box_3\neg p_3)$ (the children's simultaneous reply that they don't know whether or not they are muddy). We now sequentially update T_1 by ϕ_1 and then by ϕ_2 three times. Note that since all children are muddy, they will have to speak three times before everyone knows he is muddy.

Consider the algorithm of definition 3.2 and T_1. Following the algorithm, the refined tree $T_1 * \phi_1 = T_2$ in figure 7 is T_1 in which the links to states where no children are muddy have been removed. $T_2 * \phi_2$ is then achieved by isolating worlds that do not see two worlds (itself and another one) for every relation. In fact, only in this case one of the formulas $\Diamond_i p_i \wedge \Diamond_i \neg p_i$ can fail on a point of T_2. We can now obtain T_3 (again shown in figure 7), and T_4 similarly.

Having made all the refinements, we can now check whether or not the muddy children know to be muddy. This involves checking

$$T_4 \models p_i \Rightarrow K_i p_i.$$

Since T_4 is a singleton and $\bigwedge_{i=1}^{3} p_i$ is true at the root of T_4, the formulas are satisfied.

Analogously we can prove that the procedure given in 3.2 produces solutions for the other cases of the muddy children.

5 Properties of refinement on Kripke trees

In the rest of the paper we analyse some more properties of the refinement procedure that we defined in definition 3.12.

The first remark that we should make is that refining a scenario by some agent's knowledge cannot affect other agents' knowledge as it happened in example 2.1 for Kripke models. This is because by unravelling a Kripke model we produce a tree whose leaves are in a bijection with paths of the original model. We formalise this as follows:

Theorem 5.1 Let T be a tree, and ϕ, ψ two formulas. Then:

$$T \models \Box_i \phi \text{ implies } (T * \Box_j \psi \models \Box_i \phi), \text{ with } i \neq j.$$

[1] According the the notion of most general model as described in section 3.2 the model M should actually be $M = (2^{\{p_1,p_2,p_3\}}, U, \pi, w)$, where U is the universal relation on $W \times W$, and $\pi(w) = \{p_1, p_2, p_3\}$. The model M_1 we analyse is the result of the update of M by

$$C(p_i \Rightarrow K_j p_i) : i \neq j; i, j \in \{1,2,3\},$$

where the formula above represents the fact that children can see each other. For brevity (as in [FHMV95, HV91]) we start our analysis from M_1; i.e. rather than building the tree for M and update it first by $C(p_i \Rightarrow K_j p_i)$, we directly build the tree for M_1. The reader can check that this leads to the same result.

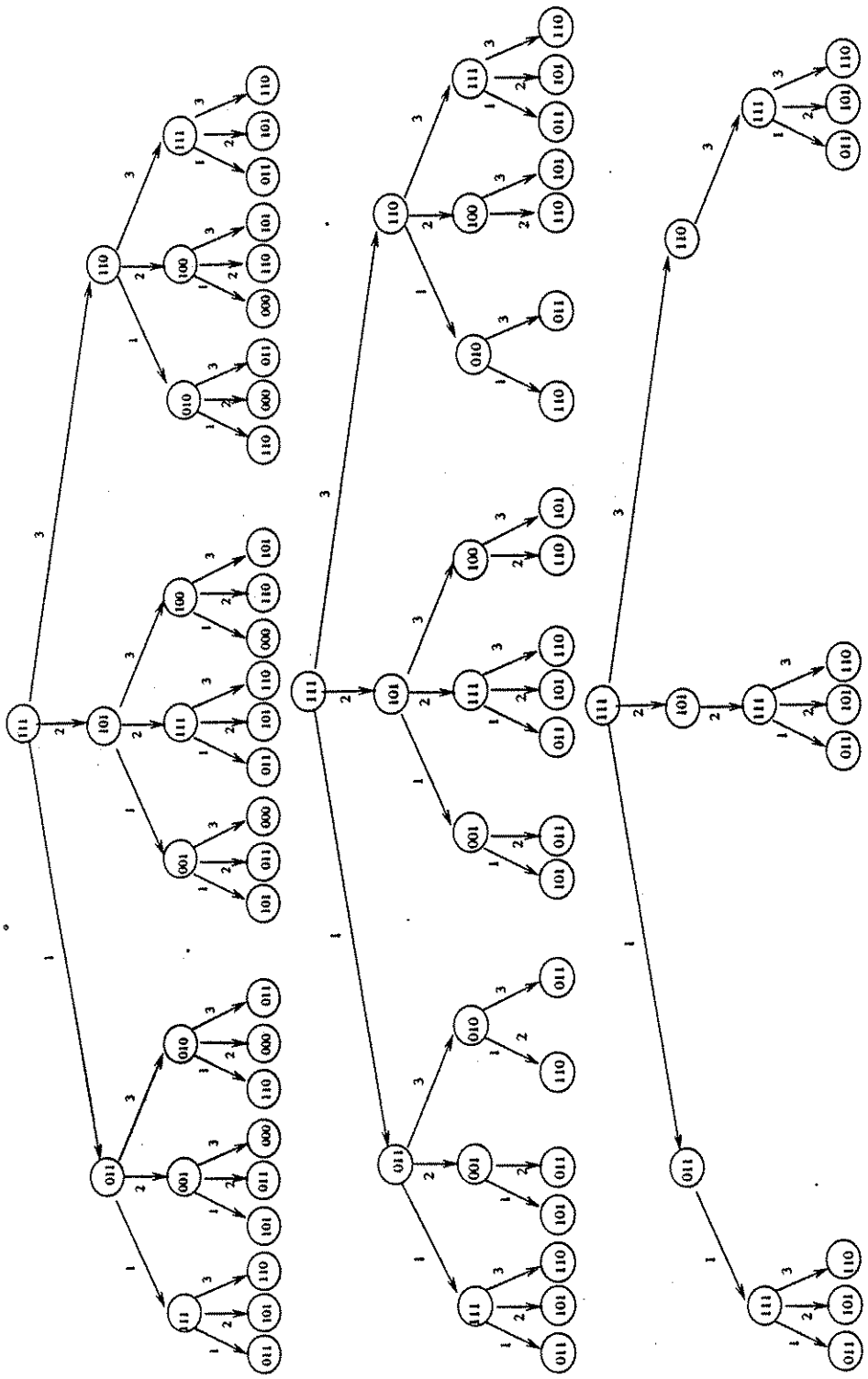


Figure 7: T_1, T_2, T_3: The Kripke trees before and after the children speak the first and the second time.

Proof Nodes of a Kripke tree are in a bijection with paths of the generating model. Therefore by removing some j-links we cannot affect the interpretation of any modality whose index is not j. □

Note that this would not be the case had we considered reflexive trees.

Although the theorem above refers to infinite trees, an analogue version can be proved for truncated trees. In that case we need the rank of the formulas to be less or equal to the depth of the truncated tree minus 1.

The second point worth stressing is that Kripke trees solve the problem of example 2.3, i.e. we can prove commutativity although the result is limited to *safe* formulas:

Definition 5.2 A formula is safe if it is universal and no $\Box_i$ and no C appear in the scope of $\vee$.

We need a few results before proving commutativity of safe formulas.

Lemma 5.3 Let ϕ, α be any formula and v any point of T.

1. $(T, v) * \phi \leqslant T$.
2. If α is universal with no disjunctions, $T_1 \leqslant T_2$ implies $(T_1, v) * \alpha \leqslant (T_2, v) * \alpha$, where $v \in V_1 \cap V_2$.
3. If α is universal then $T \models \alpha$ and $\bot \neq T' \leqslant T$ imply $T' \models \alpha$.

Proof

1. The procedure for obtaining $(T, v) * \phi$ only removes links.
2. Induction on α. We assume all negations pushed inwards. Let $T_1' = (T_1, v) * \alpha$ and $T_2' = (T_2, v) * \alpha$. Suppose α is of the form:
   - p. If $p \in \sigma(v)$ then $T_1' = T_1$, $T_2' = T_2$; else $T_1' = T_2' = \bot$.
   - $\neg p$. Similar.
   - $\beta \wedge \gamma$.
     $$\begin{aligned} (T_1, v) * \alpha &= (T_1, v) * \beta \sqcap (T_1, v) * \gamma \\ &\leqslant (T_2, v) * \beta \sqcap (T_2, v) * \gamma \quad \text{IH} \\ &= (T_2, v) * \alpha \end{aligned}$$
   - $\Box_i \beta$. Set $T_1' = T_1$ and $T_2' = T_2$ and we execute the loops of definition 3.12 ($\Box_i$-case) synchronously. We will show that $T_1' \leqslant T_2'$ is an invariant of the execution.
     Suppose $(v, v') \in E_{2i}$.
     - If $(v, v') \in E_{1i}$, then consider the following cases:
       * $(T_1, v') * \beta = \bot$ and $(T_2, v') * \beta = \bot$.
         $T_1' := T_1'|_{V - \{v'\}}$ and $T_2' := T_2'|_{V - \{v'\}}$, so $T_1' \leqslant T_2'$ is not violated.
       * $(T_1, v') * \beta = \bot$ and $(T_2, v') * \beta \neq \bot$.
         $T_1' := T_1'|_{V - \{v'\}}$ and $T_2' := (T_2', v') * \beta$, and $T_1' \leqslant T_2'$.
       * $(T_1, v') * \beta \neq \bot$ and $(T_2, v') * \beta = \bot$.
         Contradicts hypothesis that $T_1' \leqslant T_2'$.
       * $(T_1, v') * \beta \neq \bot$ and $(T_2, v') * \beta \neq \bot$.
         $T_1' := (T_1', v') * \beta$, $T_2' := (T_2', v') * \beta$, and $T_1' \leqslant T_2'$ by induction hypothesis.

- If If $(v,v') \notin E_{1i}$ then T_1' is unchanged by the body of the loop, while T_2' becomes one of $T_2' := T_2'|_{V-\{v'\}}$ and $(T_2', v') * \beta$. In either case, we are removing links in T_2 which are not present in T_1, so $T_1' \leqslant T_2'$ is preserved.

- $C\beta$. Similar to $\Box_i\beta$.

3. Induction on α.

□

Theorem 5.4 (Success) If α is universal, $(T,v) * \alpha = \bot$ or $(T,v) * \alpha \models_v \alpha$.

Proof Induction on α. The cases $\alpha = p, \neg p, \psi \vee \chi$ are straightforward; the case $\alpha = \psi \wedge \chi$ requires part 3 of lemma 5.3. □

Lemma 5.5 If α is safe, then the outcome of $(T,v) * \alpha$ is deterministically defined.

Proof Suppose ϕ contains no $\Box_i, C$. Then it's an easy induction to see that $(T,v) * \phi$ is either T or $\bot$. Now consider $(T,v) * (\phi \vee \psi)$, where ϕ, ψ are $\Box_i, C$-free. We see that either $(T,v) * \phi \leqslant (T,v) * \psi$ or $(T,v) * \psi \leqslant (T,v) * \phi$, so the result is again T or $\bot$. □

We show that, for universal formulas, the change made be a refinement is the minimal one possible in order to satisfy the formula:

Theorem 5.6 If α is safe, $(T,v) * \alpha$ is $\leqslant$-maximum in $\{T' \leqslant T \mid T' \models_v \alpha \text{ or } T' = \bot\}$.

Proof Let $T' = (T,v) * \alpha$. By part 1 of lemma 5.3 and theorem 5.4, we know T' is in the set. To prove that it is maximum, take any T'' in the set; we will show $T'' \leqslant T'$. If $T'' = \bot$ the result is immediate; otherwise, we have $T'' \models_v \alpha$ and $T'' \leqslant T$. Since $T'' \leqslant T$, we get $(T'', v) * \alpha \leqslant (T,v) * \alpha$ by part 2 of lemma 5.3. But $(T'', v) * \alpha = T''$ (since $T'' \models_v \alpha$) and $(T,v) * \alpha = T'$, so $T'' \leqslant T'$. □

Theorem 5.7 If α, β are safe $(T,v) * \alpha * \beta$ is maximum in $\{T' \leqslant T \mid T' \models_v \alpha \wedge \beta \text{ or } T' = \bot\}$.

Proof Let $T' = (T,v) * \alpha * \beta$. By parts 1 and 3 of lemma 5.3 and theorem 5.4, we know T' is in the set. The argument that it is maximum is similar to the proof of theorem 5.6. Take any T'' in the set; we will show $T'' \leqslant T'$. If $T'' = \bot$ the result is immediate; otherwise, we have $T'' \models_v \alpha \wedge \beta$ and $T'' \leqslant T$. Since $T'' \leqslant T$, we get $(T'', v) * \alpha * \beta \leqslant (T,v) * \alpha * \beta$ by part 2 of lemma 5.3. But $(T'', v) * \alpha * \beta = T''$ (since $T'' \models_v \alpha$, $(T'', v) * \alpha = T''$, and since $T'' \models_v \beta$, $(T'', v) * \beta = T''$), and $(T,v) * \alpha * \beta = T'$, so $T'' \leqslant T'$. □

Theorem 5.8 (Commutativity) If α, β are safe, then $T * \alpha * \beta = T * \beta * \alpha$.

Proof By theorem 5.7, $T * \alpha * \beta$ and $T * \beta * \alpha$ are maximum in the same set. Therefore they are equal. □

We conjecture that the premise can be relaxed to universal formulas. It is worth mentioning an example of which non-universal formulas can make commutativity to fail.

Example 5.9 Commutativity can fail for arbitrary formulas. The problem is that if the formulas are non-universal, the order of updating can play a role in the outcome of the update and we might have that one of the two cases fail. We are so far unable to find examples in which the two updates succeed but produce different result (we conjecture this is impossible; see also conclusions about this). The example we report here is the tree T_5, illustrated in figure 8, where the root is the top vertex. Consider now $T_6 = T_5 * \Diamond_1 \neg p * \Box_1(p \vee \neg q)$, illustrated, and $T_7 = T_5 * \Box_1(p \vee \neg q) * \Diamond_1 \neg p = \bot$.

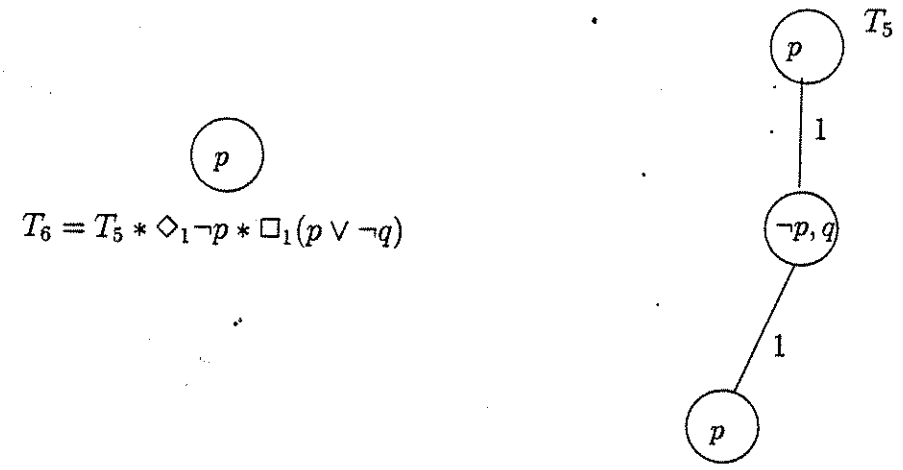


Figure 8: T_5 and T_6 discussed in example 5.9. While $T_6 = T_5 * \Diamond_1 \neg p * \Box_1(p \vee \neg q)$ is defined and shown above, $T_7 = T_5 * \Box_1(p \vee \neg q) * \Diamond_1 \neg p$ is undefined.

6 Conclusions and further work

In [Woo97] Michael Wooldridge, discussing the problems of using possible worlds semantics as formal specification for MAS, writes:

> [... possible worlds semantics are generally *ungrounded.* That is, there is usually no precise relationship between the abstract accessibility relations that are used to characterise an agent's state and any concrete computational model... this makes it difficult to go from a formal specification of a system in terms of beliefs, desires, and so on, to a concrete computational system...] (page 9)

This is indeed very often the case and this line of research aims at bringing us a step towards the use of possible worlds semantics as specification and reasoning tool for MAS.

In this paper we have developed the proposal in [HV91] for model refinement and model checking. We argued that model refinement could not be defined satisfactorily on Kripke models, and proposed a definition on Kripke trees obtained from Kripke models instead.

The shift from Kripke models to Kripke trees let us achieve two main results. First, we showed that it is possible to refine trees by a formula expressing knowledge of a formula without affecting the knowledge of the other agents (theorem 5.8) - this was not apparently possible on standard Kripke models (see example 2.1). Secondly, while it seems impossible to obtain commutativity for even safe formulas on Kripke models, we showed this is possible for Kripke trees.

Many of the issues we discussed in this note still need investigating. The following is a list of conjectures that we have not proved (or refuted) yet.

1. If $T * \phi * \psi \neq \bot$, and $T * \psi * \phi \neq \bot$, then $T * \phi * \psi \equiv T * \psi * \phi$.
2. If ϕ, ψ are universal (not necessarily safe) then $T * \phi * \psi \equiv T * \psi * \phi$. This is implied by 1.

3. Let $T = M_0$, where M_0 is the most general model and $\phi_1, \ldots, \phi_n$ be $S5^n$-consistent formulas. Then $T * \phi_1 * \ldots * \phi_n \neq \bot$. (Item 2 implies they would commute).

4. $\phi \equiv \psi$ implies $T * \phi \equiv T * \psi$.

Further work will be focused on proving the above and trying to address the following more general issues:

- What is the appropriate level of truncation that we need apply? Is there a mathematical formula that can compute it?
- Although every generated tree will satisfy $S5^n$, an update of it in general will not. Is this a strong point against model refinement as it is defined here?
- We have proved that model refinement satisfies the properties like success (theorem 5.4), commutativity (theorem 5.8). Are there other important properties that we should check or advocate, for example the ones discussed in [Gär88] and [KM91]?

References

[FHMV95] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge.* MIT Press, Cambridge, 1995.

[Gär88] P. Gärdenfors. *Knowledge in Flux: Modelling the Dynamics of Epistemic States.* MIT Press, 1988.

[Gol87] R. Goldblatt. *Logics of Time and Computation, Second Edition, Revised and Expanded,* volume 7 of *CSLI Lecture Notes.* CSLI, Stanford, 1992 (first edition 1987). Distributed by University of Chicago Press.

[HC84] G. E. Hughes and M. J. Cresswell. *A Companion to Modal Logic.* Methuen, London, 1984.

[HC96] G. E. Hughes and M. J. Cresswell. *A new introduction to modal logic.* Routledge, New York, 1996.

[Hin62] J. Hintikka. *Knowledge and Belief, an introduction to the logic of the two notions.* Cornell University Press, Ithaca (NY) and London, 1962.

[HV91] Joseph Halpern and Moshe Vardi. *Model checking vs. theorem proving: a manifesto,* pages 151–176. Artificial Intelligence and Mathematical Theory of Computation. Academic Press, Inc, 1991.

[KM91] Hirofumi Katsuno and Alberto O. Mendelzon. Propositional knowledge base revision and minimal change. *Artificial Intelligence,* 3(52):263–294, December 1991.

[Woo97] M. Wooldridge. Agent-based software engineering. *IEE Proceedings Software Engineering,* 144(1):26–37, 1997.

Reasoning by Evidence for Best-Chance Planning with Incomplete Information

Eliezer L. Lozinskii *

Abstract. *Reasoning by evidence* is presented as a way of reasoning with incomplete information. This approach is applied to the process of planning under uncertainty that stems mainly from the general incompleteness of an agent's knowledge about the real world, and from the fact that the world changes not only due to the agent's actions, but for many other reasons, often unknown to the agent.

1 Introduction

Contemplating future behavior is an essential part of activity of any intelligent agent, either a living creature or a machine. So, planning presents an important direction of research in AI. In general, given an initial state σ_0 of the agent's world W, and a goal G, a plan π is a set of actions $\{\alpha\}$ ordered either fully (linear plans) or partially (nonlinear plans [2, 6, 27, 28]) such that their execution according to this order achieves the goal. The world W of the agent is described formally by a theory S (usually in a first-order language) including a set of general constraints on the agent's activity, and a set AA of his/her/it admissible actions. Every action $\alpha \in AA$ is associated with specific preconditions (that must be satisfied to make α executable) and effects (that take place in W after a successful execution of α).

Given a specific σ_0 and G, there may not exist a way of reaching G from σ_0 by a certain agent. But often there are several possible plans for reaching the goal, constituting a set Π of plans. Plans of Π may differ in many aspects, such as the cost of their execution, duration, side effects, etc. So, an agent selects for execution a certain plan $\pi \in \Pi$ according to his preference criteria. These criteria may change in time or be dictated by the current state of the world. So, for the same σ_0 and G, different plans can be chosen by the same agent in different situations.

An important factor affecting the process of planning is an uncertainty associated with an execution of a plan. This uncertainty stems mainly from the general incompleteness of the agent's knowledge about the real world, and from the fact that the world changes not only due to the agent's actions, but for many other reasons, often unknown to the agent. Thus, before any actual execution of a planned action certain information about the current state of the world must be available or acquired to make sure that the action's preconditions are satisfied and the rest of the plan is feasible, or to chose an alternative plan, otherwise. This observation suggests producing *conditional plans* [7, 15, 17, 26] representing branching processes such that at certain points of execution a proper course of action is determined according to run-time values of conditioning parameters. The latter are measured by means of an appropriate sensing [17]. However, the state of the world at the time of execution may be different (in an unpredictable way) from that at the time when the plan has been designed. So, an important problem arises:

> What should be a rational choice of a plan in circumstances where either unavailability of a necessary information or some other uncontrolled reason hinders a predesigned execution?

2 Delayed execution

Let us consider a plan π (for reaching G from σ_0) designed at a time t_d. An agent decides to start an execution of π at a time $t_e > t_d$, however, finds out that there are some temporary obstacles to an immediate execution of π, in particular, preventing execution of an action α of π by falsifying its preconditions or hindering a proper branching by making a necessary sensing impossible. These obstacles, $Obst(\pi)$, were not anticipated when π was designed, have risen after t_d, and are supposed to be removed at some future time, after t_e. Facing this situation, the agent has to consider the following modes of behavior:

(i) Wait in the initial state σ_0 until the obstacles are removed, and then start execution of π.

(ii) Start π at t_e despite the obstacles in hope that they will be removed by the time the execution reaches α. Otherwise, the process will stop at α, wait, and proceed as soon as α becomes executable.

(iii) Abandon π, and try another plan for reaching G.

Waiting according to (i) has several drawbacks. If G has to be reached at a certain time, it may happen that if the agent waits at σ_0, then he/she/it is most likely to be late. Furthermore, when the obstacles to α are removed, some other action of π preceding α may become non-executable, so, the agent will be forced to keep waiting. Besides, the agent may wish to arrive at G as soon as possible, in the first place. Hence, the main choice is between (ii) and (iii). To choose between different plans the agent must be able to estimate the chance that a plan π will succeed given the obstacles $Obst(\pi)$. Thus,

*Inst. of Computer Science, The Hebrew University, Jerusalem 91904, Israel, email: lozinski@cs.huji.ac.il

given a set Π of previously designed plans, a rational agent should prefer a plan $\pi \in \Pi$ possessing a highest posterior probability $prob(G|S \cup \pi \cup Obst(\pi))$ that started at t_e from σ_0 it will achieve the goal G given the world description S and the obstacles $Obst(\pi)$ (expected to be removed in the future).

3 Evidence *versus* probability

Approaches to estimating and maximizing a posterior probability are based on the *Bayes' Theorem* that allows computing unknown probability of some event given corresponding probabilities of other ones [16, 24]. Let $p(A)$, $p(A|B)$ denote, respectively, the prior probability that a statement A is true, and the posterior probability that A is true given that B is true. If $p(A) \neq 0$ and $B_1, B_2, \ldots, B_n$ are mutually exclusive and jointly exhaustive statements such that $\sum_{i=1}^{n} p(B_i) = 1$, then

$$p(B_i|A) = \frac{p(B_i)p(A|B_i)}{p(A)} = \frac{p(B_i)p(A|B_i)}{\sum_{i=1}^{n} p(B_i)p(A|B_i)}.$$

To apply the Bayesian approach to computing of $p(B_i|A)$ one needs to know $p(B_i)$ and $p(A|B_i)$ for all $i = 1, 2, \ldots, n$. This requirement of a full specification of probability distribution is often difficult to meet in real systems.

Several methods have been developed to cope with a lack of statistical information. A celebrated one is the *Principle of Maximum Entropy* [11, 12, 35]. The notion of entropy, widely used in physics, was considered by Hartley [9] and Wiener [37] in connection with information transmission, and introduced into Information Theory by Claude Shannon [30]. Let P be a probability distribution assigning probabilities $p_1, \ldots, p_n$ to n mutually exclusive and jointly exhaustive events (or statements) of a set R. Then the *entropy* of P, $H(P)$, is defined by

$$H(P) = -\sum_{i=1}^{n} p_i \log p_i, \qquad (1)$$

where $p_i \log p_i$ is assigned zero if $p_i = 0$.

$H(P)$ is a measure of uncertainty associated with R given the distribution P. In particular, $H(P)$ reaches its maximum value of $\log n$ if all the events of R are equiprobable,

$$p_1 = \ldots = p_n = 1/n. \qquad (2)$$

This distribution is, indeed, most uncertain, since it does not allow to expect one event more than any other, or to prefer the truth of some statement of R to that of another one.

Now suppose that some knowledge K is available about real relationships existing among the elements of R. This knowledge can be regarded as a set of constraints on R. As the available knowledge about the real world is usually incomplete, there exists a set $\{P_K\}$ of different distributions satisfying the constraints. Then an important question is, which one of the various P_K is most adequate to the reality? This *preferred distribution*, P_K^*, must reflect the fact that there is no other knowledge about R beyond that presented by K. So, besides satisfying the constraints, P_K^* must maximize the uncertainty associated with $R \cup K$. And this is just the decision suggested by the Principle of Maximum Entropy: Given R and K, prefer a probability distribution P_K^* satisfying K such that for all P_K, $H(P_K^*) \geq H(P_K)$. This principle, which can be traced back some 100 years, to the works of Boltzmann, Maxwell, Gibbs, and Planck, and has been introduced in its modern form by Jaynes [11] and Tribus [35], proved its efficiency in various fields of Physics, Statistics, Information Theory, Pattern Recognition, Signal Processing, etc. [1, 8, 11, 12, 13, 14, 18, 25, 32, 33, 34, 35, 38].

To be applied successfully, most of the known approaches to reasoning with uncertain and incomplete information require an extensive statistical knowledge (e.g., prior and posterior probabilities for Bayesian methods, basic probability assignment in Dempster-Shafer theory [3, 4, 29], prior beliefs for computing certainty factors in MYCIN [10, 31], partial probability assignments for Nilsson's Probabilistic Logic [23]). In many practical situations such statistical data are either only partially available or insufficiently reliable. These circumstances raise an important question of *how reasoning with incomplete information can be performed efficiently in the absence of reliable statistical data.*

Although the available knowledge S may not imply the truth or falsehood of a statement F, it provides some *semantic information* about F. As S enables reaching certain conclusions regarding the truth of F, we may say that S provides certain *evidence* of F (denoted $E(S, F)$). We would like the value of evidence $E(S, F)$ to present a measure of credibility of a belief in the truth of F.

To present a measure of credibility of a belief in the truth of F, the value of evidence has to possess the following reasonable properties:

(1) If F is a logical consequence of S, then the truth of F is most credible, so, $E(S, F|S \models F) = 1$. (Let the range of evidence be normalized in [0, 1]).

(2) If $S \models \neg F$, then the belief in the truth of F has no credibility.
Hence, $E(S, F|S \models \neg F) = 0$.

(3) In particular, if ϕ, ψ are a tautology and an unsatisfiable formula, respectively, then for all S, $E(S, \phi) = 1$ and $E(S, \psi) = 0$.

(4) $E(S, \phi \vee \psi) = E(S, \phi) + E(S, \psi) - E(S, \phi \wedge \psi)$.

(5) Properties (3) and (4) imply for all S and ψ, $E(S, \psi) + E(S, \neg\psi) = 1$.

As evidence of F provided by S is intended to replace the unavailable probability of truth of F, the properties (1)–(5) of evidence resemble the axioms of probability. However, evidence differs from probability in several ways. Conceptually, evidence, unlike probability, is based on the semantics of a logic system, rather than on the notion of chance, repeated trials, experience or subjective estimates. Quantitatively, evidence is different from probability, but it accommodates the available statistical information, and amounts to the probability if this information becomes sufficiently large (see Section 4).

If S represents a world W faithfully, then every momentary state of W corresponds to a model of S, although,

because of an incompleteness of the knowledge presented in S, some models of S may describe states of W that do not occur in the reality. However, these states appear to be possible from the standpoint of the current knowledge. This observation underlies the widely adopted view of W as a set of states, referred to as *possible worlds*, such that there is a bijection between the set of possible worlds and the set $MOD(S)$ of all models of S (S is supposed to be consistent).

At any moment the world is in one and only one of its possible states. So, $MOD(S)$ provides a set of mutually exclusive and exhaustive particular representations of W. Let $p(\mu)$ denote the probability that W is in a state represented by a model $\mu \in MOD(S)$. Then the probability $p(F)$ that a formula F is true in the current state of W can be expressed as

$$p(F) = \sum_{\mu \in MOD(S)} p(F|\mu)p(\mu) = \sum_{\mu \in MOD(S) \wedge \mu \models F} p(\mu), \tag{3}$$

since

$$p(F|\mu) = \begin{cases} 1 & if \ \ \mu \models F \\ 0 & otherwise. \end{cases}$$

Unfortunately, a full probability distribution over the space of possible states is usually either not available or not satisfactorily reliable, although a partial distribution may be known for some possible worlds represented by a subset of models $M \subset MOD(S)$. However, if M does not contain all models asserting F, then a question arises of how the probability of the models not contained in M should be estimated.

4 Applying the Principle of Maximum Entropy

Consider a user interested in finding out the actual state of a world W given its description by a knowledge system S. As the system represents every possible world by means of one of its models, a probability distribution over $MOD(S)$ is necessary for reasoning about the actual state of W. Suppose that prior probabilities $q(\mu)$ are known for models of a subset M of $MOD(S)$. Given this partial distribution P_M, a rational reasoner should prefer a full distribution P_{MOD} over $MOD(S)$ that (a) is consistent with S, (b) includes P_M, and (c) implies no extra knowledge beyond S and P_M. According to the Principle of Maximum Entropy [11, 35], P_{MOD} must maximize the value of Shannon's entropy $H(P)$ (expression (1)). This implies the following definition of P_{MOD}:

$$p(\mu) = \begin{cases} q(\mu) & if \ \ \mu \in M \\ \frac{1-\sum_{\mu \in M} q(\mu)}{|MOD(S)-M|} & otherwise. \end{cases} \tag{4}$$

Now the evidence $E(S, F)$ can be defined as the probability that F is true in the current state of the world given the reasonable, although not necessarily actual, distribution P_{MOD} (expression (4)).

Definition 4.1 (Evidence) *Due to expressions (3) and (4),*

$$E(S,F) = \sum_{\mu \in M \wedge \mu \models F} q(\mu) + \sum_{\mu \in (MOD(S)-M) \wedge \mu \models F} p(\mu)$$

$$= \sum_{\mu \in (M \cap MOD(S \cup \{F\}))} q(\mu)$$

$$+(1 - \sum_{\mu \in M} q(\mu)) \frac{|MOD(S \cup \{F\}) - M|}{|MOD(S) - M|}, \tag{5}$$

where $|MOD(S)|$, $|MOD(S \cup \{F\})|$ denote, respectively, the cardinality of the set $MOD(S)$ of all models of S, and that of its subset consisting of all models in which F is true.

If $MOD(S \cup \{F\}) \subseteq M$, then $E(S, F)$ amounts to the probability of F. If no prior probabilities of possible worlds are known, such that $M = \emptyset$, then

$$E(S,F) = \frac{|MOD(S \cup \{F\})|}{|MOD(S)|}. \tag{6}$$

Proposition 4.1 *Definition 4.1 is in accord with properties (1)–(5) of evidence (Section 3).*

Proof is omitted for brevity.

An implicit assumption underlying Definition 4.1 is that the set of models of S is finite. However, the value of evidence can be calculated in systems with infinite sets of models [21].

5 Reasoning by evidence about a best-choice plan

The preceding considerations suggest the following way of reasoning by adopting evidence $E(S, F)$ as a measure of credibility of a belief in the truth of F given S.

Definition 5.1 (Reasoning by evidence) *(i) If $E(S, F) = 1$ (or $E(S, F) = 0$), then F is true (false, respectively);*

(ii) If $E(S, F) > 0.5$ (that is, $E(S, F) > E(S, \neg F)$), then F is more likely to be true than false, and the larger the value of $E(S, F)$ the more credible a belief in the truth of F; If $E(S, F) = 0.5$, then both possible beliefs of F have the same (rather low) credibility.

(iii) If $E(S, F) > E(S, F')$, then the truth of F is more credible than that of F'.

Let us recall the problem of choosing a best-chance plan, that is, a plan $\pi^* \in \Pi$ possessing a highest chance of achieving the goal, given the incomplete knowledge available at time t_e, namely, that there are obstacles to a full execution of the plans of Π that are expected to be removed after t_e, although no information regarding the time or likelihood of the removal.

Let $\vec{\pi}$ denote a fragment of π not affected by $Obst(\pi)$ consisting of all actions of π that can be performed according to the information available at t_e. Then reasoning by evidence suggests the following way of choosing a best-chance plan.

Definition 5.2 (Best-chance plan) *Given a world description S, a set Π of plans achieving a goal G, and a description of obstacles $Obst(\pi)$ for all $\pi \in \Pi$, then a plan $\pi^* \in \Pi$ is a* best-chance plan *if an execution of its fragment $\vec{\pi}^*$ provides G with a highest evidence:*

$$E(S \cup \vec{\pi}^*,\ G) = \max_{\pi \in \Pi} [E(S \cup \vec{\pi},\ G)] .$$

6 An example

Let us consider a simplistic motivating example of an electric circuit displayed in Figure 1.

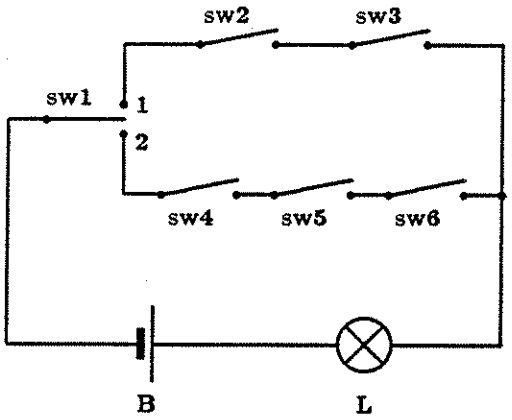


Figure 1. A sample circuit.

It is composed of a light bulb L, battery B, and switches $sw1, \ldots, sw6$. Normally the battery and the bulb are operational, but all the switches are in state *off*, so, the bulb is not lit. There are two plans to lit L: either put $sw1$ in state *1* and $sw2, sw3$ in state *on* (so, $\pi_1 = \{On1(sw1),\ On(sw2),\ On(sw3)\}$), or put $sw1$ in state *2* and $sw4, sw5, sw6$ in state *on* ($\pi_2 = \{On2(sw1),\ On(sw4),\ On(sw5),\ On(sw6)\}$). Suppose that $sw1, \ldots, sw6$ are located in different places, but all the switches are equally accessible to an operator. To lit the bulb (that is the goal) the operator would prefer to operate three switches $sw1, sw2, sw3$ rather than four ones $sw1, sw4, sw5, sw6$. Now suppose that $sw2, sw3$ and $sw6$ become (temporarily) inaccessible to the operator. It appears reasonable that in this case the operator would put $sw1$ in state *2* and switch on $sw4, sw5$ in hope that it is more likely to get an access to a single switch $sw6$ than to two inaccessible ones $sw2, sw3$. Besides, it is possible that some of the inaccessible switches are already switched on, and again, it is more likely that a single $sw6$ is *on* than the two, $sw2$ and $sw3$, are so.

Now let S include a description of the circuit, and an assumption that the battery and the bulb are normal. We wish to light the bulb, that is, make $G = On(L)$. If only $sw1$, $sw4$ and $sw5$ are accessible, then the unaffected fragment of π_1 and π_2 are, respectively, $\vec{\pi}_1 = \{On1(sw1)\}$ and $\vec{\pi}_2 = \{On2(sw1),\ On(sw4),\ On(sw5)\}$. To determine a best-chance plan, let us compute the evidence of $On(L)$ provided by $\vec{\pi}_1$ and $\vec{\pi}_2$. Table 1 gives values of evidence $E(S \cup \{\phi\}, On(L))$ for all possible combinations ϕ of states of the accessible switches (if $sw1$ is *off* then obviously the bulb cannot be lit, so, $E(S \cup \{\phi\}, On(L)) = 0$ for all ϕ containing $Off(sw1)$). The first and second lines of the table correspond to $\vec{\pi}_1$ and $\vec{\pi}_2$, respectively. It follows that π_2 is the best-chance plan, since it provides an evidence of G twice as high as π_1 does. And the rest of the table shows that $\vec{\pi}_2$ is better than any other possible operation of the accessible switches. Thus, the plan suggested by the reasoning by evidence conforms to the operator's common sense decision justified above.

Table 1. Possible actions.

$M1 = |MOD(S \cup \{\phi\})|$, $M2 = |MOD(S \cup \{\phi\} \cup On(L))|$
$E = E(S \cup \{\phi\}, On(L))$

π	ϕ			$M1$	$M2$	E
	sw1	sw4	sw5			
$\vec{\pi}_1$	On1	Off	Off	8	2	0.25
$\vec{\pi}_2$	On2	On	On	8	4	**0.50**
	On1	On	On	8	2	0.25
	On1	On	Off	8	2	0.25
	On1	Off	On	8	2	0.25
	On2	On	Off	8	0	0.00
	On2	Off	On	8	0	0.00
	On2	Off	Off	8	0	0.00

7 Counting models

Reasoning by evidence requires computing number of models of logic formulas which is a hard computational task. Valiant [36] has shown that in general the problem is #P-complete. Dubois [5] and Zhang [39] have developed algorithms for computing the number of models of a propositional CNF formula (with n variables, m clauses, k literals in each clause) having a worst case time complexity of $O(mr_k^n)$, where r_k is the largest root of $x^k - x^{k-1} - \ldots - x - 1$ (such as $r_2 = 1.618$, $r_3 = 1.839$, $r_4 = 1.928, \ldots$, $\lim_{k \to \infty} r_k = 2$.

Very hard in general, the problem of counting models turns out to be quite tractable in many practical cases. Luby and Velickovic [22] have presented an algorithm for approximating the number of models of a DNF propositional formula with m clauses on n variables. The running time of the algorithm is estimated by a polynomial in m and n multiplied by $n^{O(\frac{1}{\delta}(\log^2 m + \log^2 \frac{1}{\epsilon})(\log\log m + \log\frac{1}{\epsilon} + \log\frac{1}{\delta}))}$, where ϵ, δ are the approximation errors. Linial and Nisan [19] have presented an algorithm for computing the size of a union of a family of sets, that can be used for approximating the number of models. The algorithm runs in time $O(m^c)$, where c grows with the precision. It has been shown in [20] that under reasonable assumptions the average run time of computing the exact number of models of a propositional formula is $O(m^d n)$ where $d = O(\log m)$.

References

[1] Aczel, J. and Forte, B. Generalized Entropies and the Maximum Entropy Principle. In *Maximum Entropy and Bayesian Methods in Applied Statistics*. Ed. by J. H. Justice (Cambridge University Press, 1986) 95-100.

[2] Chapman, D. Planning for conjunctive goals. *Artificial Intelligence* **32** (1987) 333-377.

[3] Dempster, A. P. Upper and lower probabilities induced by a multivariate mapping. *Annals of Mathematical Statistics*, v. 38, 1967, 325-339.

[4] Dempster, A. P. A generalization of Bayesian inference. *J. of the Royal Statistical Society*, v. 30, 1968, 205-247.

[5] O. Dubois. Counting the number of solutions for instances of satisfiability. *Theoretical Computer Science*, v. 81, 1991, 49-64.

[6] Eshghi, K. Abductive planning with event calculus. *Proc. 5th International Conference and Symposium on Logic Programming*, R. Kowalski and K. Bowen, Eds., Seattle, Wash., August 1988, MIT Press, 1988, 563-579.

[7] Etzioni, O., Hanks, S., Weld, D., Draper, D., Lesh, N. and Williamson, N. An approach to planning with incomplete information. *Proc. 3rd International Conference on Principles of Knowledge Representation and Reasoning*, October 1992, Cambridge, Mass., Morgan Kaufmann, San Mateo, CA, 1992, 115-125.

[8] Goldszmidt, M., Morris, P., and Pearl, J. A maximum entropy approach to nonmonotonic reasoning. *Proc. 8th Natl. Conf. on Artificial Intelligence, AAAI'90*, Boston, Mass., July 29 - August 3, 1990, The MIT Press, Menlo Park, CA, 646-652.

[9] Hartley, R. V. Transmission of information. *Bell Systems Technical Journal*, v.7 (1928) 535-563.

[10] Heckerman, D. Probabilistic interpretation of MYCIN's certainty factors. In *Uncertainty in Artificial Intelligence*. Ed. by L. N. Kanal and J. F. Lemmer (North Holland, 1986) 167-196.

[11] Jaynes, E. T. Information Theory and Statistical Mechanics. *Physical Review*, v. 106 (1957) 620-630.

[12] Jaynes, E. T. Where do we stand on maximum entropy? In *The Maximum Entropy Formalism*. Ed. by R. D. Levine and M. Tribus (Cambridge, MA: MIT Press, 1979) 15-118.

[13] Jaynes, E. T. Where do we go from here? In *Maximum-Entropy and Bayesian Methods in Inverse Problems*. Ed. by C. R. Smith and W. T. Grandy (Reidel Publishing Co., Dordrecht, Holland, 1985) 21-58.

[14] Justice, J. H., Ed. *Maximum Entropy and Bayesian Methods in Applied Statistics* (Cambridge University Press, 1986).

[15] Krebsbach, K., Olawsky, D., Gini, M. An empirical study of sensing and defaulting in planning. *Proc. 1st Conference on AI Planning Systems*, San Mateo, CA, 1992, 136-144.

[16] Lee, N. S., Grize, Y. L., and Dehnad, K. Quantitative models for reasoning under uncertainty in Knowledge-Based Expert Systems. *International J. of Intelligent Systems*, v. 2 (1987) 15-38.

[17] Levesque, H. J. What is planning in the presence of sensing? *Proc. 13th National Conference on Artificial Intelligence*, August 1996, Portland, Oregon, AAAI Press, Menllo Park, CA, 1996, 1139-1146.

[18] Levine, R. D. and M. Tribus, M., Eds. *The Maximum Entropy Formalism* (MIT Press, 1979).

[19] Linial, N., and Nisan, N. Approximate inclusion-exclusion. *Combinatorica*, v.10, no. 4 (1990) 349-365.

[20] Lozinskii, E. Computing propositional models. *Information Processing Letters*, v. 41 (1992) 327-332.

[21] Lozinskii, E. Is there an alternative to parsimonious semantics? *J. of Experimental and Theoretical Artificial Intelligence*, v. 6 (1994) 163-193.

[22] Luby, M., and Velickovic, B. On deterministic approximation of DNF. *Proc. 23rd ACM Symp. on Theory of Computing*, New Orleans, LA, May 1991, ACM Press, 430-438.

[23] Nilsson, N. J. Probabilistic logic. *Artificial Intelligence* 28 (1986) 71-87.

[24] Pearl, J. *Probabilistic Reasoning in Intelligent Systems* (Morgan Kaufmann, Los Altos, CA, 1988).

[25] Pearl, J. Probabilistic semantics for nonmonotonic reasoning: a survey. *Proc. First Intl. Conf. on Principles of Knowledge Representation and Reasoning* (Toronto, May 1989) Morgan Kaufmann, San Mateo, CA, 1989, 699-710.

[26] Peot, M. and Smith, D. Conditional nonlinear planning. *Proc. 1st Conference on AI Planning Systems*, San Mateo, CA, 1992, 189-197.

[27] Sacerdoti, E. D. The nonlinear nature of plans. *Proc. IJCAI-75*, Tbilisi, U.S.S.R., 1975, 206-214.

[28] Sacerdoti, E. D. *A Structure of Plans and Behavior* (American Elsevier, New York, 1977).

[29] Shafer, G. *A Mathematical Theory of Evidence* (Princeton University Press, Princeton, NJ, 1976).

[30] Shannon, C. E. A mathematical theory of communication. *Bell Systems Technical Journal*, v. 27 (1948) 379-423, 623-656.

[31] Shortliffe, E. H. *Computer-Based Medical Consultation: MYCIN* (Elsevier, 1976).

[32] Skyrms, B. Maximum entropy inference as a special case of conditionalisation. *Synthese*, v. 63 (1985) 55-74.

[33] Smith, C. R. and G. J. Erickson, Eds. *Maximum-Entropy and Bayesian Spectral Analysis and Estimation Problems* (Reidel Publishing Co., Dordrecht, Holland, 1983).

[34] Smith, C. R. and W. T. Grandy, Jr. (eds.) *Maximum-Entropy and Bayesian Methods in Inverse Problems* (Reidel Publishing Co., Dordrecht, Holland, 1985).

[35] Tribus, M. Information Theory as the basis for Thermostatics and Thermodynamics. *J. of Applied Mechanics*, v. 28 (1961) 1-8.

[36] Valiant, L. The complexity of computing the permanent. *Theoretical Computer Science* 8 (1979) 189-201.

[37] Wiener, N. *Cybernetics, or Control and Communication in the Animal and the Machine* (Wiley, New York, 1948).

[38] Williams, P. M. Bayesian conditionalisation and the principle of minimum information. *British J. for the Philosophy of Science*, v. 31, 1980, 131-144.

[39] Zhang. W. Number of models and satisfiability of sets of clauses. *Theoretical Computer Science*, v. 155, 1996, 277-288.

The Dialectic of Practical Reasoning

Munindar P. Singh
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206, USA
singh@ncsu.edu

Abstract

The Webster Collegiate Dictionary defines the "dialectic" as a method of intellectual investigation that relies on discussion and reasoning by dialogue. Socrates used the dialectic as a technique for exposing false beliefs and eliciting truth. The dialectic also applies to any systematic reasoning, exposition, or argument that juxtaposes opposed and contradictory ideas and usually seeks to resolve their conflict.

For our purposes, the dialectic is all of the above, as well as a technique whereby a number of autonomous interacting agents may reason practically *together*. The field of practical reasoning has studied a number of metaphors and approaches using which intelligent (and, of necessity, limited) agents may cope with a complex and rapidly changing world. The dialectic provides a natural metaphor for situations where the agents must reason together.

The dialectic leads naturally to a series of arguments exchanged among the agents. However, the arguments not only help resolve conflicts or identify commonalities, but also lay the ground for further debate and reasoning. Agents working individually would concentrate their energies on finding evidence or answering questions that arise in their ongoing interactions. In this way, argumentation has the usual dilemma of practical reasoning approaches. On the one hand, it helps focus reasoning; on the other hand, it can lead to suboptimal solutions. However, unlike the single-agent approaches, the artifacts of argumentation are inherently public. Thus, they can help a group of agents collectively satisfice with the confidence that as individuals they would be exonerated for doing no more than following what the rest of their group is interested in.

Recognizing the centrality of the dialectic is the first step in developing specialized approaches for collective practical reasoning. This

talk is geared more toward drawing attention to this area than toward offering solutions.

Deliberate Robbery, or the Calculating Samaritan

Leendert W.N. van der Torre[1] and Yao-Hua Tan[2]

Abstract. In this paper we introduce the deliberating robber, an example of reasoning about preferences in a logic of desires. We show that two defeasible reasoning schemes proposed in qualitative decision theory derive counterintuitive consequences.

1 Introduction

In the usual approaches to planning in AI, a planning agent is provided with a description of some state of affairs, a *goal state*, and charged with the task of discovering (or performing) some sequence of actions to achieve that goal. Context-sensitive goals are provided for situations in which the agent encounters goals that it cannot achieve, for example when its objectives can be satisfied to varying degrees, such as time taken to fill a tank or amount of fluid spilled. The desirability aspect of context-sensitive goals is formalized with preference-based or utilitarian semantics, which are provided by decision theory [8, 10, 3].[3] Besides expressing the desirability of a state, adopting a goal represents some commitment to pursuing that state [8]. However, we do not discuss this second aspect of goals in this paper.

Recently, several more or less *ad hoc* preference-based logics for goals and desires have been proposed [10, 9, 27, 3, 31, 30, 17, 34, 35], which raises a demand for criteria to evaluate the properties of the logics. In this paper we introduce to this end the deliberating robber, a relative of the gentle murderer. The formalization of both persons is problematic, because they specify different degrees of sub-ideal behavior. They differ in two respects. First, the deliberating robber is subject to decision-theoretic instead of moral norms. However, we argue that this distinction is irrelevant for the problem at hand. Second, the deliberating robber is described in more detail than the gentle murderer. There is an indefinite number of distinct degrees of robbing (instead of two degrees of killing) and there is a distinction between contexts. Moreover, the number of degrees or contexts may change (instead of keeping it fixed). The deliberating robber example is used to analyze the behavior of two preference-based logics developed in qualitative decision theory, and we show that counterintuitive conclusions follow.

[1] IRIT, Paul Sabatier University, 118 Route de Narbonne, 31062 Toulouse, France, torre@irit.fr

[2] EURIDIS, Erasmus University Rotterdam, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands, ytan@fac.fbk.eur.nl

[3] Much of decision theory is concerned with conditions under which the preference ordering is representable by an order-preserving, real-valued *value* or (under uncertainty) *utility function*, and with identifying regularities in preferences that justify value or utility functions with convenient structural properties [21].

2 The gentle murderer

In 1984, James Forrester wrote a paper called "Gentle murder, or the adverbial Samaritan" [11]. It was soon recognized that the gentle murderer example introduced in that paper is the most notorious paradox of monadic deontic logic – Castañeda called it 'the deepest paradox of all' [5] –, because standard techniques developed to formalize related problems introduced two decades before [7, 1] could not be used to formalize the example. This notorious example consists of the following four sentences, which are intuitively consistent. The four sentences are inconsistent in so-called Standard Deontic Logic[4], and it is therefore called a paradox.

$O\neg k$	Smith should not kill Jones.
$k \rightarrow Og$	If Smith kills Jones, then he should do it gently.
k	Smith kills Jones, and
$\vdash g \rightarrow k$	Gentle killing logically implies killing.

The problem of the paradox is the formalization of the different degrees in which the norm not to kill can be violated. There are usually several degrees of violating a norm in criminal law, with different sanctions associated with them. For example, a legal system associates different sanctions to different degrees of robbery, like robbing a bank accompanied with using violence, taking hostages, killing people, etc. They are all violations of the norm not to rob, but they violate it in different degrees. In the style of the gentle murderer paradox they can be formalized by 'Smith should not rob Jones,' $O(\neg r|\top)$, 'if Smith robs Jones, then he should not use violence,' $O(\neg v \mid r)$ 'if Smith robs Jones and he uses violence, then he should not kill Jones,' $O(\neg k|v \wedge r)$ etc. In this example the law describes the different degrees of violating a norm. However, the norm 'the harder you drive, the worse your violation (and the higher your penalty)' does not describe the different degrees, and it is up to the modeler to characterize what counts as a substantial difference. In deontic logic it is assumed that normally it makes sense to stipulate different degrees of violation, and talk about them with contrary-to-duty preferences, although the borders between different degrees may be vague, and often several partitions are possible.

The gentle murderer paradox is called a contrary-to-duty paradox, or *the* contrary-to-duty paradox, because the second obligation is a so-called contrary-to-duty obligation of the first obligation, which means that the second obligation is only in

[4] Standard Deontic Logic is a monadic modal logic of type KD according to the Chellas classification [6]. It is the smallest set that that it is closed under the inference rules modus ponens $\frac{\beta \rightarrow \alpha, \beta}{\alpha}$ and necessitation $\frac{\vdash \alpha}{\vdash O\alpha}$, and contains the propositional tautologies and the axioms **K**: $O(\alpha \rightarrow \beta) \rightarrow (O\alpha \rightarrow O\beta)$ and **D**: $\neg(O\alpha \wedge O\neg\alpha)$.

L.W.N. van der Torre and Y.-H. Tan
PRR 98. *3rd Workshop on Practical Reasoning and Rationality*
Workshop of the
ECAI 98. *13th European Conference on Artificial Intelligence*

force if the first obligation is violated. Formally, the conditional obligation $\beta \rightarrow O\alpha$ is a contrary-to-duty (or secondary) obligation of the (primary) obligation $O\alpha_1$ if and only if $\beta \wedge \alpha_1$ is inconsistent. An additional requirement of the formalization of contrary-to-duty reasoning is the so-called pragmatic oddity introduced in [28]. It consists of the following three sentences.

Op	You should keep your promise,
$\neg p \rightarrow Oa$	If you have not kept your promise, then you should apologize,
$\neg p$	You have not kept your promise.

The second sentence is a contrary-to-duty obligation of the first obligation, because its condition – not keeping your promise – is a violation of the first sentence. In Standard Deontic Logic the pragmatic oddity is consistent, in contrast to the gentle murderer, but the counterintuitive 'you ought to keep your promise and apologize for not keeping it' $O(p \wedge a)$ can be derived. It is therefore also called a paradox.

The most flexible formalization of both examples, originating from [15, 23], is to represent obligations by *dyadic* operators. A dyadic obligation $O(\alpha|\beta)$ is read as 'α ought to be (done) if β is (done).' The gentle murderer is formalized by $\{O(\neg k \mid \top), O(g \mid k), \vdash g \rightarrow k, k\}$ and the pragmatic oddity by $\{O(p|\top), O(a|\neg p)\}$. The problem of dyadic deontic logic is which axioms can be added to it. For example, factual detachment $O(\alpha|\beta) \wedge \beta \rightarrow O\alpha$ cannot be added, or the paradoxes are directly reinstated. For example, the contradictory $O\neg k$ and Og can be derived from $O(\neg k|\top)$, $O(g|k)$ and k with this axiom.

The semantics of the Hansson-Lewis logic contains value structures, usually loosely called preference orderings. Assume a standard possible worlds model $M = (W, \leq, V)$ with a totally connected, transitive and reflexive accessibility relation $\leq$, that does not contain infinite descending chains. We have $M \models O_{HL}(\alpha|\beta)$ if the preferred β worlds are α worlds, i.e. for all β-worlds w_1 such that there is not a β-world w_2 with $w_1 \leq w_2$ and $w_2 \not\leq w_1$, we have $M, w_1 \models \alpha$. A typical preference-based model of the gentle murderer paradox is represented in Figure 1 [33]. This figure should be read as follows. The circles are equivalence classes of worlds and the arrows represent strict preferences for all worlds in the circles. The transitive closure is left implicit. The worlds represented by a circle satisfy the propositions in the circle. The two dyadic obligations $O_{HL}(\neg k \mid \top)$ and $O_{HL}(g \mid k)$ state that the best worlds are $\neg k$ worlds, and the best k worlds are g worlds. Hence, they give rise to at least three different states: no killing, gentle killing and brutal killing.

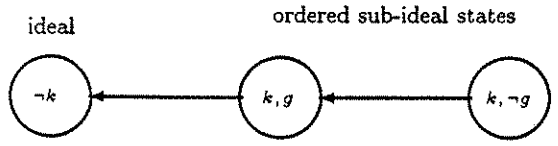


Figure 1. Model of the gentle murderer paradox

A drawback of the Hansson-Lewis logic is that it does not have strengthening of the antecedent. For example, the obligation that 'Smith should not kill Jones when it is raining' $O_{HL}(\neg k|r)$ cannot be derived from the obligation 'Smith should not kill Jones' $O_{HL}(\neg k|\top)$. This is called the irrelevance problem, because the logic cannot deal with irrelevant facts like raining. In the semantics the problem is that there can be more than three different states, whereas the two obligations seem to justify only *exactly* the three given states. In the default logic literature it is well known that the irrelevance problem can be solved by non-monotonic techniques. In fact, nearly all preference-based default logics can be used to generate exactly the three states of the gentle murderer paradox represented in Figure 1. However, this does not mean that preference-based default logics can be used as deontic logics. Whereas all preference-based default logics deal well with the gentle murderer, they run into problems with the deliberating robber, which is introduced in this paper.

3 The deliberating robber

The deliberating robber has given preferences about robbing a bank, which she has calculated taking into account the utilities and probabilities of success and failure of the robbery. For example, she dislikes the sanctions when caught, but she likes the money she gains if the robbery is successful, and maybe she also likes the attention in the press, the thrill etc. She differs in two respects from the gentle murderer. First, she is subject to decision-theoretic instead of moral norms. We write $D(\alpha|\beta)$ for 'the agent desires (to do) α if β is (done).' The desires discriminate between ideal and ordered sub-ideal states, just like the gentle murderer in Figure 1. However, the distinction is not made by a moral or legal normative system that promulgated norms, but by the agent itself. It is subjective. Second, she is described in more detail than the gentle murderer. We consider more than two distinct degrees of robbing a bank, and we consider different contexts of robbing. In this paper the intended reading of the different circumstances or contexts is that they stand for different places, although they could be read as other types of context too. For example, we discriminate between robbing a bank in the United States and robbing a bank in Europe. Contexts are represented by propositional sentences which appear in the antecedent of the contrary-to-duty preferences. For example, 'Smith desires not to rob a bank in the context where she is in the Netherlands' is formalized by $D(\neg r|n)$.

We introduce some special notation, because otherwise the figures and formulas tend to be rather difficult to read. The mutually exclusive different degrees of robbing are written as $r_1, r_2, r_3, \ldots$, where we have $\vdash \neg(r_i \wedge r_j)$ for $i \neq j$. Degree 1 is the least desired degree of robbing, degree 2 is the second least desired, etc. Likewise, the distinct and mutually exclusive contexts are written as $c_1, c_2, \ldots$. In this paper we only consider two contexts, and we simplify notation to c and $\neg c$ instead of c_1 and c_2. Finally, we assume in our examples that the order of the degrees of robbing is the same in each context. In context c as well as context $\neg c$ we have that r_1 is the least desired, then r_2, etc. Obviously, this is unrealistic, and contradicts the need of using contexts. However, it makes the figures easier to read, and it is a trivial exercise to show that our analyses in this paper do not depend on this assumption.

Example 1 (The deliberating robber) *Consider preferences about robbing, represented in Figure 2. This figure*

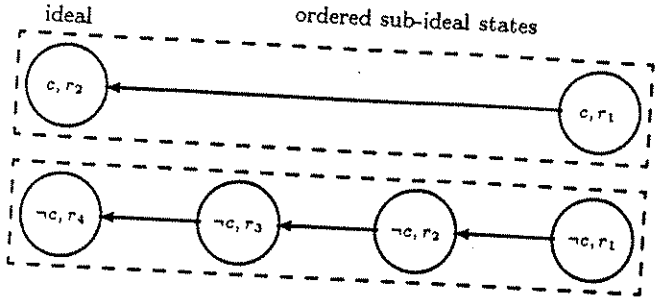


Figure 2. Robbery preferences

should be read as follows. It consists of two partial models of respectively c and ¬c worlds, represented by two dashed boxes. The partial models in the dashed boxes are analogous to the model of the gentle murderer in Figure 1. In context c there are two degrees of robbing and in context ¬c there are four degrees of robbing. We assume all propositions are controllable, in the terminology of [9], thus the robber can control whether she is in context c or in context ¬c and she can control the degree of her robbery. Notice that we do not assume any preferences for either c or ¬c. The preferences for when the robber is in context c and when she is in context ¬c are divided in two (unrelated) groups. It is unclear how the c and ¬c worlds should be combined, and we show later that this is the main source of the counterintuitive derivations made in several preference-based logics.

Now consider the situation in which the context c preferences are specified more precisely, as represented in Figure 3. In context c, we distinguish six different degrees instead of a

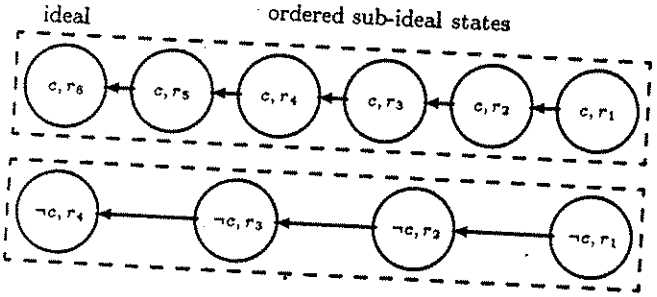


Figure 3. Revised robbery preferences

binary distinction. Intuitively, this revised specification should not influence the derivable preferences, except for the preferences that concern robbery in context c. In particular, it is highly counterintuitive if the first specification implies a preference for context c, and the second specification a preference for context ¬c (or vice versa).

In this paper we formalize the deliberating robber in two preference-based logics developed in qualitative decision theory, and we show that counterintuitive conclusions follow. However, first we compare the deliberating robber with the gentle murderer.

4 Comparison

The gentle murderer is a person formalized by deontic logic, because he is subject to moral norms. The deliberating robber is formalized by decision theory, because she is subject to decision-theoretic norms. If she is rational, then she is maximizing her utility. Surprisingly, the gentle murderer and the deliberating robber are related. In our comparison we only consider preferences related to utilities, and we do not discuss the complexities introduced by probabilities and uncertainty. First, the preference-based semantics of deontic logic has been interpreted as a utilitarian semantics [20, 27]. Second, decision logic [19] uses a representation of utilities which is closely related to preference-based deontic logics. In the qualitative decision theory literature, the relation between qualitative decision theory and deontic logic is described as a structural similarity from different perspectives.

Structural similarity There is a structural similarity between logics for desires (or goals) in qualitative decision theory and deontic logic, because both are based on preferences [3, 22, 32]. In qualitative decision theory preferences are used to formalize *context-sensitive* goals [10], and in deontic logic preferences are introduced to formalize reasoning in the context of violations, that is, contrary-to-duty reasoning.

Different perspectives The main purpose of a deontic logic is deriving new obligations (and permissions) from an initial specification, while qualitative decision theory focuses on the search for optimal acts and decisions [22]. Deontic logic and decision logic have different perspectives. Obligations are exogenous (they are imposed by a legal or moral code) while desires in decision logics are endogenous (coming from the agent) [22]. It is this distinction which we call *the gap between desires and obligations.*[5]

After the introduction of the gentle murderer we mentioned that different degrees of robbing also occur in legal (or deontic) reasoning. As a consequence of the structural similarity, the deliberating robber can be analyzed as a decision-theoretic problem as well as a deontic problem. This intuition is made explicit by the following two assumptions.

Assumption 1 *The gap between desires and obligations is irrelevant for the formalization of the gentle murderer and the deliberating robber.*

Motivation *Consider an autonomous robot and its rational (in the sense of decision theory) owner, and perfect communication between them. Moreover, assume a rational environment in which the autonomous robot normally does what it is obliged to do. Hence, we exclude irrealistic circumstances in which for example the robot does exactly what is forbidden. Under these assumptions, the rational owner makes sure that the robot is obliged to do what she wants it to do, because this*

[5] The gap does not exist for robots, who interpret the obligations of their owner as their goals, i.e. as their desires. That is how the (non-autonomous!) robot is programmed: the normative code is regimented in the robot. McCarty [24] calls this regimentation the causal assumption, with which it is possible to reason about the real world with the use of deontic logic. It implements the 'innocent until proven' assumption, because it assumes that an obligation is fulfilled when it can be either fulfilled or violated.

maximizes her expected utility. In other words, the obligations of an autonomous robot are a reflection of the desires of its owner. If we reason about these preferences, we can call the qualitative expressions obligations (from the perspective of the robot) or desires (from the perspective of the owner).

If the deliberating robber is not a typical deontic or decision-theoretic problem, then what is it? We now argue that it is a problem of combining preferences.

Assumption 2 *When we accept the preference-based solution of the Forrester paradox, then the deliberating robber is a problem of combining preferences.*

Motivation *First consider the extension to an indefinite number of degrees of violating a norm. This does not introduce any new problems, because formalizations of the deliberating robber by the partial models in Figure 2 and 3 are as good as the formalization of the gentle murderer by the model in Figure 1. Second, consider the extension to different contexts. This is just the problem of combining the two dashed boxes in Figure 2 and 3, which is a problem of combining preferences.*

The deliberating robber extends the gentle murderer, because it has an indefinite number of degrees and contexts, and the number of degrees or contexts may change. It is important to analyze the sensitivity of the formalization to the number of degrees or contexts, because these numbers are often arbitrary, i.e. several partitions are equally plausible. In the following section we show that two defeasible reasoning schemes proposed in qualitative decision theory are sensitive to the number of degrees of violating a norm.

5 Testing the deliberating robber

In this section we test the logics introduced in [3] and [31] on the deliberating robber. Both do not deal satisfactorily with contrary-to-duty preferences, because the schemes force (almost) unique totally connected orderings. Lack of incomparable worlds turn these schemes into violation counters, in the sense that under certain circumstances they can only discriminate between the number of violations.

5.1 Boutilier

Boutilier develops a logic of qualitative decision theory in which the basic concept of interest is the notion of *conditional preference*. Boutilier writes $I(\alpha|\beta)$, read "ideally α given β," to indicate that the truth of α is preferred, given β. Boutilier proposes an extension of the Hansson-Lewis logic O_{HL} with Pearl's System Z [26] to deal with the irrelevance problem. System Z is a popular way to solve the irrelevance problem of conditional logic, and it is equivalent to Lehmann's Rational Closure and the so-called minimal specificity principle of possibilistic logic. Moreover, several other well-known defeasible reasoning schemes are extensions of System Z in the sense that they derive a superset of the conclusions of System Z [12, 14]. System Z adds strengthening of the antecedent by assuming that 'worlds gravitate towards ideal.' Here we give the reconstruction of gravitating towards ideal of Boutilier [2]. The basic idea is that worlds are more preferred in preference relation $\leq_1$ than in $\leq_2$ when they are equivalent to a more preferred world. The definitions are slightly complicated because worlds can be not equivalent to any other world: condition (2.), and to allow that $N(w, M_1, M_2)$ is false. Given the definition of more preferred worlds, Boutilier defines a preference ordering $\sqsubseteq$ on models. The preferred models are used for preferential entailment [29].

Definition 1 *A preference model $M = (W, \leq, V)$ is a possible worlds model with a reflexive, transitive and totally connected accessibility relation $\leq$.*

Definition 2 *Let M be a modal preference model. We have $M \models I(\alpha|\beta)$ iff there is a world w_1 such that $M, w_1 \models \alpha \wedge \beta$ and all worlds $w_2 \leq w_1$ we have $M, w_2 \models \beta \rightarrow \alpha$.*

Definition 3 (System Z) *Let $M_1 = (W, \leq_1, V)$ and $M_2 = (W, \leq_2, V)$ be two modal preference models with the same W and V. Let T be a set of dyadic preferences $I(\alpha|\beta)$. $w \in W$ is more preferred in M_1 than in M_2, written as $N(w, M_1, M_2)$, iff*

1. *there is some $v \in W$ such that $v \leq_1 w$, $w \leq_1 v$, and not $v \leq_2 w$, or*
2. *there is no v (with $w \neq v$) such that $w \leq_2 v$ and $v \leq_2 w$.*

The model M_1 is as preferable as M_2, written as $M_1 \sqsubseteq M_2$, iff for all $w \in W$, $N(w, M_1, M_2)$ is false only if $\{v \mid w \leq_2 v\} \subseteq \{v \mid w \leq_1 v\}$. M_1 is preferred to M_2, written as $M_1 \sqsubset M_2$, iff $M_1 \sqsubseteq M_2$ and $M_2 \not\sqsubseteq M_1$. M is a preferred model of T iff $M \models T$ and for all M' such that $M' \models T$, we have $M' \not\sqsubset M$. α is preferentially entailed by T, written as $T \models_{\sqsubset} \alpha$, iff $M \models \alpha$ for all preferred models M of T.

Although the definition of system Z is quite complex, the underlying mechanism is quite simple, as shown by the following example. It also illustrates that gravitating towards the ideal derives counterintuitive consequences for the deliberating robber.

Example 2 (Deliberating robber, continued) *The preferences of Example 1 are formalized by the following two sets S and S'.*

$$S : \left\{ \begin{array}{l} I(r_2|c), I(r_4|\neg c), I(r_3|\neg c \wedge \neg r_4) \\ I(r_2|\neg c \wedge \neg r_4 \wedge \neg r_3) \end{array} \right\}$$

$$S' : \left\{ \begin{array}{l} I(r_6|c), I(r_5|c \wedge \neg r_6), I(r_4|c \wedge \neg r_6 \wedge \neg r_5) \\ I(r_3|c \wedge \neg r_6 \wedge \neg r_5 \wedge \neg r_4) \\ I(r_2|c \wedge \neg r_6 \wedge \neg r_5 \wedge \neg r_4 \wedge \neg r_3), I(r_4|\neg c) \\ I(r_3|\neg c \wedge \neg r_4), I(r_2|\neg c \wedge \neg r_4 \wedge \neg r_3) \end{array} \right\}$$

The unique System Z models of S and S' are represented in Figure 4. For example, in the System Z model of S, the $c \wedge r_2$ and $\neg c \wedge r_4$ worlds are equivalent, as well as the $c \wedge r_1$ and $\neg c \wedge r_3$ worlds.

Unfortunately, we have $S \models_{\sqsubset} I(c|r_1)$, because the preferred r_1 worlds are only the $c \wedge r_1$ worlds. This is counterintuitive, because intuitively the most ideal states for context c are the $c \wedge r_1$ worlds, and the most preferred worlds for context $\neg c$ are the $\neg c \wedge r_1$ worlds, and these worlds are incomparable. Gravitating towards ideality prefers context c, because it contains only one violation of a desire, whereas the latter contains three violations. However, the first desire may be more

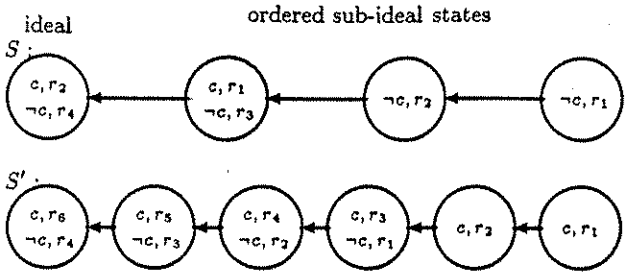


Figure 4. Gravitating towards the ideal (System Z)

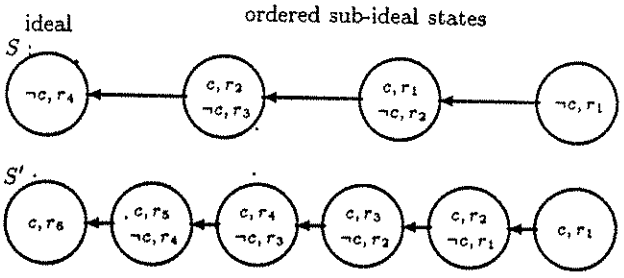


Figure 5. Gravitating towards the center (compactness)

desirable than the latter three desires together. Moreover, consider the revised degrees of robbery. We have $S' \models_{\sqsubset} I(\neg c|r_1)$. *Hence, because we have further specified the degrees of robbery in context 1, now we have the desire to be in context* $\neg c$ *instead of context* c. *Obviously, this is highly counterintuitive.*

5.2 Tan and Pearl

Tan and Pearl [31] propose a logic of ceteris paribus preferences, together with a gravitation mechanism in which there is no preference for either end of the bipolar preference scale. In this paper we do not discuss the ceteris paribus preferences, because they are irrelevant for our formalization of the deliberating robber, and we focus on the defeasible reasoning scheme (which is also used in [30]). Instead of a preference for the ideal, there is a preference for the center. From the set of admissible preference rankings they select the ranking which minimizes the difference in the preference ranks. The following definition is based on preferential entailment, analogous to the definition of System Z.

Definition 4 *A ranked possible worlds model* $M^r = \langle W, \pi, V\rangle$ *contains a ranking* π, *a function from worlds to integers. Its preference model* $M = \langle W, \leq, V\rangle$ *is given by* $w_1 \leq w_2$ *iff* $\pi(w_1) \leq \pi(w_2)$, *and we say that* M^r *satisfies a formula if its preference model satisfies the formula. A ranked model* M_1^r *is at least as compact as another ranked model* M_2^r *iff*

$$\Sigma_{w_i,w_j \in W}(|\pi_1(w_i) - \pi_1(w_j)|) \leq \Sigma_{w_i,w_j \in W}(|\pi_2(w_i) - \pi_2(w_j)|)$$

We write $\models_{TP}$ *for preferential entailment based on the most compact models.*

The following example illustrates the deliberating robber for Tan and Pearl's compactness rule.

Example 3 (Speed limits, continued) *Reconsider the set of preferences of Example 2. The unique preference model of the most compact models of* S *and* S' *is given in Figure 5. We have* $S \models_{TP} I(\neg c|\top)$: *there is a preference for context* $\neg c$. *With revised speed limits we have* $S' \models_{TP} I(c|\top)$: *there is a preference for context* c. *Again, this is highly counterintuitive.*

The problem of the logics of Boutilier and Tan and Pearl is the combination of an (almost) unique model and a totally connected ordering. Consider the unique preferred models in Figure 4 and 5. Some c and $\neg c$ worlds are forced to be equivalent, whereas intuitively they are incomparable. As a consequence, the contrary-to-duty paradoxes arise.

6 Further research

A remarkable distinction between the two logics considered in this paper is that the first is monopolar (it only considers the bad violation pole) whereas the second is bipolar (it considers the bad violation pole and the good ideal pole). Tan and Pearl [31] criticize the approach of gravitating towards ideality, because 'while it is intuitive to assume that worlds gravitate towards normality because abnormality is a monopolar scale, it is not at all clear that worlds ought to be as preferred as possible since preference is a bipolar scale.' In a preference logic, there are good ideal and bad violation poles. Boutilier [3] defends his monopolar approach: 'Note that in classical decision theory, such distinctions do not exist. An outcome cannot be good or bad, nor can an agent be indifferent toward an outcome, in isolation; it can only be judged *relative* to other outcomes. An agent can adopt an attitude towards a *proposition*.' We presently investigate the impact of polarity in preference-based logics [35].

We argued that for the gentle murderer and deliberating robber examples, it does not matter whether we formalize them in qualitative decision theory and deontic logic as a consequence of the structural similarity. However, these two theories are quite distinct. As a consequence of the gap between desires and obligations, we distinguish the following types of questions. The first type of questions is raised in qualitative decision theory literature and the latter two types are raised in deontic logic literature.

1. Can a deontic logic be used as a logic of goals or desires in qualitative decision theory? If so, which type of deontic logic can be used?
   Ad 1 The structural similarity suggests that preference-based deontic logic can be used in a qualitative decision theory to formalize reasoning about goals or desires (the assumption of this paper). However, in the preference logic literature, it has been questioned whether there is a general theory of preference underlying different applications. For example, Mullen [25] concluded that a logic of preference can only be tested in the process of testing the more general theory in which it is embedded, in this case qualitative decision theory or deontic logic.
2. Can deontic logic be based on decision-theoretic concepts?
   Ad 2 Jennings [20] proposed that deontic logic could be based on utilitarian semantics and Pearl [27] proposed a decision-theoretic account of obligation statements that uses qualitative abstractions of utilities and probabilities.

Pearl used his logic of pragmatic obligations to criticize deontic logic by arguing that 'exploratory reading of the literature reveals that philosophers hoped to develop deontic logic as a branch of conditional logic, not as a synthetic amalgam of logic of belief, action, and causation,' and that 'such an isolationistic strategy has little chance of succeeding.' This criticism is in our opinion not very convincing, because deontic logic is characterized by the distinction between actual and ideal, not by beliefs and actions. Finally Pearl concludes that 'the decision-theoretic account can be used to generate counterexamples to most of the principles suggested in the literature.' However, his logic is in our opinion better understood as a defeasible deontic logic, because it also formalizes uncertainty. Obviously, *any* defeasible deontic logic can be used to generate 'counterexamples' for a non-defeasible deontic logic.

3. Is qualitative decision theory an extension of deontic logic in the sense that it formalizes reasoning with norms? Is there a role for norms in qualitative decision theory?
Ad 3 Deontic logic has been developed as a branch of philosophical logic, and it has recently been studied in computer science. Topics identified are legal knowledge-based systems, the specification of fault tolerant systems, the specification of security policies, the automatization of contracting and the specification of normative integrity constraints for databases [36]. However, deontic logic is not sufficient for all applications that are based on normative reasoning. The problem is that deontic logic only formalizes reasoning *about* obligations, that is, which obligations follow from a set of obligations. However, there is a demand to formalize reasoning *with* obligations. For example, a legal expert system may face the diagnostic problem to determine whether a suspect has violated a legal rule, and a robot may have to solve the planning problem how to fulfill the desires of his owner. It is an open problem whether the techniques developed in qualitative decision theory can be used.

7 Conclusions

In this paper we introduced the deliberating robber, a relative of the gentle murderer. We showed counterintuitive consequences for two defeasible reasoning schemes proposed in logics for context-sensitive goals or desires in qualitative decision theory. Moreover, we discussed the relation between the logic of desires and the logic of obligations. The structural similarity between the logic of desires and the logic of obligations suggests that preference-based deontic logics [20, 18, 13, 16, 4] can be used in qualitative decision theory. However, the different perspectives suggests that further research is needed.

REFERENCES

[1] L. Åqvist, 'Good Samaritans, contrary-to-duty imperatives, and epistemic obligations', *Noûs*, **1**, 361–379, (1967).
[2] C. Boutilier, 'Conditional logics for default reasoning and belief revision', Technical Report 92-1, Department of Computer Science, University of British Columbia, (1992).
[3] C. Boutilier, 'Toward a logic for qualitative decision theory', in *Proceedings of the KR'94*, pp. 75–86, (1994).
[4] A.L. Brown, S. Mantha, and T. Wakayama, 'Exploiting the normative aspect of preference: a deontic logic without actions', *Annals of Mathematics and Artificial Intelligence*, **9**, 167–203, (1993).
[5] H. Castañeda, 'Aspectual actions and the deepest paradox of deontic logic', in *Automated Aspects of Legal Texts*, eds., A.A. Martino and F. Socci Natali, 31–50, Elsevier Science Publishers B.V., (1986).
[6] B.F. Chellas, *Modal Logic: An Introduction*, Cambridge University Press, 1980.
[7] R.M. Chisholm, 'Contrary-to-duty imperatives and deontic logic', *Analysis*, **24**, 33–36, (1963).
[8] T. Dean and M. Wellman, *Planning and Control*, Morgan Kaufmann, San Mateo, 1991.
[9] J. Doyle, Y. Shoham, and M.P. Wellman, 'The logic of relative desires', in *Sixth International Symposium on Methodologies for Intelligent Systems*, Charlotte, North Carolina, (1991).
[10] J. Doyle and M.P. Wellman, 'Preferential semantics for goals', in *Proceedings of AAAI-91*, pp. 698–703, Anaheim, (1991).
[11] J.W. Forrester, 'Gentle murder, or the adverbial Samaritan', *Journal of Philosophy*, **81**, 193–197, (1984).
[12] H. Geffner and J. Pearl, 'Conditional entailment: bridging two approaches to default reasoning', *Artificial Intelligence*, **53**, 209–244, (1992).
[13] L. Goble, 'A logic of good, would and should, part 2', *Journal of Philosophical Logic*, **19**, 253–276, (1990).
[14] M. Goldszmidt, P. Morris, and J. Pearl, 'A maximum entropy approach to nonmonotonic reasoning', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **15**, 220–232, (1993).
[15] B. Hansson, 'An analysis of some deontic logics', in *Deontic Logic: Introductionary and Systematic Readings*, 121–147, D. Reidel Publishing Company, Dordrecht, Holland, (1971).
[16] S.O. Hansson, 'Preference-based deontic logic (PDL)', *Journal of Philosophical Logic*, **19**, 75–93, (1990).
[17] Z. Huang and J. Bell, 'Dynamic goal hierarchies', in *Intelligent Agent Systems: Theoretical and Practical Issues*, LNAI 1027, pp. 88–103. Springer, (1997).
[18] F. Jackson, 'On the semantics and logic of obligation', *Mind*, **94**, 177–196, (1985).
[19] R. Jeffrey, *The Logic of Decision*, University of Chicago Press, 2nd edn., 1983.
[20] R.E. Jennings, 'A utilitarian semantics for deontic logic', *Journal of Philosophical Logic*, **3**, 445–465, (1974).
[21] R.L. Keeney and H. Raiffa, *Decisions with Multiple Objectives: Preferences and Value Trade-offs*, Wiley and Sons, New York, 1976.
[22] J. Lang, 'Conditional desires and utilities - an alternative approach to qualitative decision theory', in *Proceedings of the ECAI'96*, pp. 318–322, (1996).
[23] D. Lewis, 'Semantic analysis for dyadic deontic logic', in *Logical Theory and Semantical Analysis*, 1–14, D. Reidel Publishing Company, Dordrecht, Holland, (1974).
[24] L.T. McCarty, 'Modalities over actions: 1. model theory', in *Proceedings of the KR'94*, pp. 437–448, San Francisco, CA, (1994). Morgan Kaufmann.
[25] J.D. Mullen, 'Does the logic of preference rest on a mistake?', *Metaphilosophy*, **10**, 247–255, (1979).
[26] J. Pearl, 'System Z: A natural ordering of defaults with tractable applications to default reasoning', in *Proceedings of TARK'90*, pp. 121–135, (1990).
[27] J. Pearl, 'From conditional oughts to qualitative decision theory', in *Proceedings of the UAI'93*, pp. 12–20, (1993).
[28] H. Prakken and M.J. Sergot, 'Contrary-to-duty obligations', *Studia Logica*, **57**, 91–115, (1996).
[29] Y. Shoham, *Reasoning About Change*, MIT Press, 1988.
[30] S.-W. Tan and J. Pearl, 'Qualitative decision theory', in *Proceedings of the AAAI'94*, pp. 928–933, (1994).
[31] S.-W. Tan and J. Pearl, 'Specification and evaluation of preferences under uncertainty', in *Proceedings of the KR'94*, pp. 530–539, (1994).
[32] R. Thomason and R. Horty, 'Nondeterministic action and dominance: foundations for planning and qualitative decision', in *Proceedings of the TARK'96*, pp. 229–250. Morgan Kaufmann, (1996).
[33] L.W.N. van der Torre, *Reasoning about Obligations: Defeasibility in Preference-based Deontic Logic*, Ph.D. dissertation, Erasmus University Rotterdam, 1997.

[34] L.W.N. van der Torre, 'Labeled logics of goals', in *Proceedings of the ECAI'98*, pp. 368–369. John Wiley & Sons, (1998).
[35] L.W.N. van der Torre and E. Weydert, 'Goals, desires, utilities and preferences', in *Proceedings of the ECAI'98 Workshop on Decision Theory meets Artificial Intelligence*, (1998).
[36] R.J. Wieringa and J.-J.Ch. Meyer, 'Applications of deontic logic in computer science: A concise overview', in *Deontic Logic in Computer Science*, 17–40, John Wiley & Sons, Chichester, England, (1993).

Intention Reconsideration Reconsidered

Michael Wooldridge and **Simon Parsons**
Department of Electronic Engineering
Queen Mary and Westfield College
University of London
London E1 4NS, United Kingdom
{M.J.Wooldridge, S.D.Parsons}@qmw.ac.uk

Abstract

In this paper, we consider the problem of designing agents that successfully balance the amount of time spent in deliberation (reasoning about what intentions to adopt) against the amount of time spent acting to achieve intentions. Following a brief review of the various ways in which this problem has previously been analysed, we motivate and introduce a simple formal model of agents, which is closely related to the well-known belief-desire-intention model. In this model, an agent is explicitly equipped with mechanisms for deliberation and action selection, as well as a meta-level control strategy, which allows the agent to choose between deliberation and action. Using the formal model, we define what it means for an agent to be optimal with respect to a task environment, and explore how various properties of an agent's task environment can impose certain requirements on its deliberation and meta-level control components. We then show how the model can capture a number of interesting practical reasoning scenarios, and illustrate how our notion of meta-level control can easily be extended to encompass higher-order meta-level reasoning. We conclude with a discussion and pointers to future work.

1 Introduction

There is a well-known problem in the literature of practical reasoning agents, which relates to the amount of time an agent spends in *deliberating* about what to do and the amount of time it spends *acting* [2, 6, 13]. If an agent could perform arbitrarily large computations in constant time, then it could determine the optimal action to perform at any instant by using standard decision theoretic techniques, say by considering all possible alternatives and picking the best — the one with the highest expected utility. But of course, no real agent has this capability. All real agents are *resource bounded*: they have finite computational resources, and operate under externally imposed time-constraints. So an agent has a limited amount of time in which to select an appropriate course of action. But if an agent spends insufficient time reasoning, then the quality of its decision may well be poor. This fundamental problem, (which in [16] was characterised as achieving a balance between *deliberation* and *reaction*), is perhaps the key problem faced by agent designers.

The implications of resource bounds were pointed out comparatively early in the history of artificial intelligence (AI), perhaps most notably by Simon in *The Sciences of the Artificial* [14]. Nevertheless, it has been argued that AI has been somewhat slow as a discipline to deal with these implications [13]. Historically, the predominant approach to designing agents capable of rational action has been to specify (typically in terms of first-order logic or one of its variants) an *optimal* agent, and to employ symbolic (typically logical) reasoning in order to animate this specification and so generate an agent's behaviour. But as Russell and colleagues point out [13, 12], this tradition is predicated on the assumption that the action ultimately selected by the reasoning process, which *would be* optimal *when reasoning began* is

also optimal *when reasoning concludes*. This assumption simply does not hold for any real-time or dynamic environment.

Much of the research activity from the intelligent agent community in the mid-to-late 1980s was focussed around the problem of designing agents that could effectively balance deliberation and action. In particular, a number of agent control architectures were developed that attempted to balance the amount of time spent in deliberation and action by making use of pre-compiled courses of action, whose applicability could easily be determined with respect to the agent's current situation (see [16] for a survey of such architectures). One particularly successful approach that emerged at this time was the *belief-desire-intention* (BDI) paradigm [5, 2, 9, 10].

The development of the BDI paradigm was to a great extent driven by Bratman's theory of (human) practical reasoning [1], in which *intentions* play a central role. Put crudely, since an agent cannot deliberate indefinitely about what courses of action to pursue, the idea is it should eventually *commit* to achieving certain states of affairs, and then devote resources to achieving them. These chosen states of affairs are intentions, and once adopted, they play a number of roles. For example, intentions provide a filter for future practical reasoning: once an agent has adopted an intention to φ, it need not subsequently consider any courses of action that are incompatible with φ. Moreover, once an agent has adopted an intention to φ, we expect that it will actively devote resources to achieving φ. Intentions thus provide *stability* and a *focus* for an agent's practical reasoning. Because intentions play such a key role in practical reasoning, there have been a number of attempts to develop adequate formal models of intention, of which a good example is [3].

A major issue in the design of agents that are built upon models of intention is that of when to reconsider intentions. An agent cannot simply maintain an intention, once adopted, without ever stopping to reconsider it. From time-to-time, it will be necessary to check, (for example), whether the intention has been achieved, or whether it is believed to be no longer achievable [3]. In such situations, it is necessary to *deliberate* over intentions, and, if necessary, to *change focus* by dropping existing intentions and adopting new ones. Kinny and Georgeff undertook an experimental study of different intention reconsideration strategies [6]. They found that highly dynamic environments — environments in which the rate of world change was high — tend to favour *cautious* intention reconsideration strategies, i.e., strategies which frequently stop to reconsider intentions. Intuitively, this is because although such agents incur the costs of deliberation, they do not waste effort attempting to achieve intentions that are no longer viable, and are able to exploit new opportunities as they arise. In contrast, *static* environments — in which the rate of world change is low — tend to favour *bold* intention reconsideration strategies, which only infrequently pause to reconsider intentions.

Our aim in this paper is to consider the question of when to deliberate (i.e., to reconsider intentions) *versus* when to act from a formal point of view, in contrast to the experimental standpoint of Kinny and Georgeff [6]. We develop a simple formal model of practical reasoning agents, and investigate the behaviour of this model in different types of task environment. In this agent model, (which is very closely related to the BDI model [5, 2, 9, 10]) an agent's internal state is characterised by a set of beliefs (information that the agent has about its environment) and a set of intentions (commitments the agent has made about what states of the world to try and achieve). In addition, an agent has a deliberation function, which allows it to reconsider and if necessary modify its intentions, and an action function, which allow it to act towards its current intentions. These functions are mediated by a *meta-level control* function. The purpose of the meta-level control strategy is simply to choose between deliberation and action. The meta-level control strategy thus acts somewhat like the interpreter in the PRS [5], but more closely resembles the meta-plans that are used to manage an agent's intention structures in the PRS.

The remainder of this paper is structured as follows. In section 2 we present our formal model of agents, and we define what it means for an agent to be optimal with respect to a particular *task environment*. In section 3, we formally define what it means for a task environment to be *real time*, and we investigate the relationships that must hold between an agent's meta-level control and deliberation components in order for an agent to act optimally in such task environments. In particular, we define

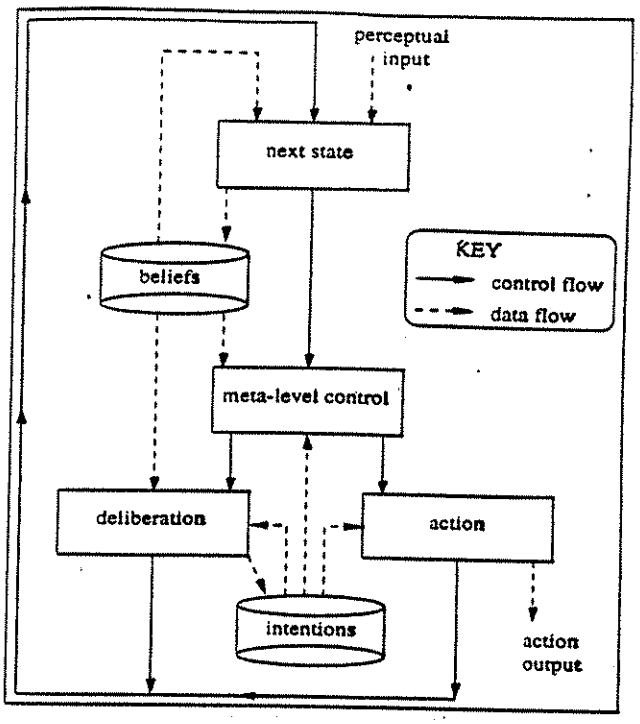


Figure 1: Meta-level control, deliberation, and action in an architecture for practical reasoning agents.

notions of soundness and completeness for meta-level control and deliberation strategies, and show that an optimal meta-level control strategy must be sound and complete with respect to a deliberation strategy in real-time task environments. In section 4, we show how our formal framework can capture a number of typical practical reasoning scenarios (taken from [2]). In section 5, we generalise our model of meta-level control to capture *higher-order* meta-level reasoning strategies (intuitively, strategies to determine what sort of meta-level reasoning strategy to use), and we integrate these with our agent model. Finally, in section 6, we present some conclusions and issues for future work.

2 Agents and Environments

In this section, we introduce the mathematical foundations of our analysis. In particular, we formalise a simple model of practical reasoning agents and the environments they occupy, and define what we mean by a *run* or *history* of an agent in an environment. An overview of our agent model is given in Figure 1.

As this figure illustrates, an agent has two main data structures: a *belief set* and an *intention set*. An agent's beliefs represent information that the agent has about its environment. In implemented agent systems (such as PRS [5]), beliefs are often represented symbolically, as PROLOG-like facts, but they may simply be variables of a PASCAL-like programming language. However they are represented, beliefs correspond to an agent's *information state*. Let B be the set of all beliefs. For the most part, the contents of B will not be of concern to us here. However, it is often useful to suppose that B contains formulae of some logic, so that, for example, it is possible to determine whether two beliefs are mutually consistent or not. An agent's actions at any given moment are guided by its *intention set*, which represents its *focus*: the "direction" of its activities. Intentions may be thought of as states of affairs that an agent has committed to bringing about. Formally, let I be the set of all intentions. Again, we are not concerned here with the contents of I. As with beliefs, however, it is often useful to assume that intentions are expressed in some sort of logical language. An agent's *local state* will then be a pair (b, i), where $b \subseteq B$ is a set of beliefs, and $i \subseteq I$ is a set of intentions. The local state of an agent is its internal state: a

snapshot of its information and focus at any given instant. Let $L = \wp(B) \times \wp(I)$ be the set of all internal states. We use l (with annotations: $l', l_1, \ldots$) to stand for members of L. If $l = (b, i)$, then we denote the belief component of l by b_l, and the intention component of l by i_l.

Agents do not operate isolation: they are situated in *environments*; we can think of an agent's environment as being everything external to the agent. (This external component may, of course, include other agents; we leave the exploration of this possibility to future work.) We assume that the environment external to the agent may be in any of a set $E = \{e, e', \ldots\}$ of states.

Together, an agent and its environment make up a *system*. The *global* state of a system at any time is thus a pair containing the state of the agent and the state of the environment. Formally, let $G = E \times L$ be the set of all such global states. We use g (with annotations: $g, g', \ldots$) to stand for members of G.

2.1 Choice, Deliberation, and Action

As Figure 1 illustrates, our agents have four main components, which together generate their behaviour: a *next-state function*, a *meta-level control function*, a *deliberation function*, and an *action function*. The *next state* function can be thought of as a *belief revision function*. On the basis of the agent's current state and the state of the environment, it determines a new set of beliefs for the agent, which will include any new information that the agent has perceived. An agent's next-state function thus realises whatever *perception* the agent is capable. Formally, a next-state function is a mapping $\mathcal{N} : E \times \wp(B) \rightarrow \wp(B)$.

The next component in our agent architecture is meta-level control. The idea here is that at any given instant, an agent has two choices available to it. It can either *deliberate* (that is, it can expend computational resources deciding whether to change its focus), or else it can *act* (that is, it can expend resources attempting to actually achieve its current intentions). Note that we assume the only way an agent can *change* its focus (i.e., modify its intentions) is through explicit deliberation. To represent the choices (deliberation versus action) available to an agent, we will assume a set $C = \{d, a\}$, where d denotes deliberation, and a denotes action. The purpose of an agent's *meta-level control strategy* it to choose between deliberation and action. If it chooses to deliberate, then the agent subsequently deliberates; if its chooses to act, then the agent subsequently acts. Formally, we can represent such strategies as functions $\mathcal{M} : L \rightarrow C$, which on the basis of the agent's internal state, decides whether to deliberate (d) or act (a).

The *deliberation* process of an agent is represented by a function which on the basis of an agent's internal state (i.e., its beliefs and intentions), determines a new set of intentions. Formally, we can represent this deliberative process via a function $\mathcal{D} : L \rightarrow \wp(I)$. As an aside, note that we require the cost of an agent's meta-level control strategy to be insignificant in comparison to the cost of its deliberation strategy. The intuition behind this requirement is discussed later.

If an agent decides to act, rather than deliberate, then it is acting to achieve its intentions. To do so, it must decide *which* action to perform. The action selection component of an agent is essentially a function that, on the basis of the agent's current state, returns an action, which represents that which the agent has chosen to perform. Let $Ac = \{\alpha, \alpha', \ldots\}$ be the set of actions. Formally, an action selection function is a mapping $\mathcal{A} : L \rightarrow Ac$.

Collecting these components together, we define an agent to be a 5-tuple $(\mathcal{M}, \mathcal{D}, \mathcal{A}, \mathcal{N}, l_0)$, where $\mathcal{M}$ is a meta-level control strategy, $\mathcal{D}$ is a deliberation function, $\mathcal{A}$ is an action selection function, $\mathcal{N}$ is a next-state function, and $l_0 \in L$ is an *initial state*.

2.2 Runs

Recall that agents are situated in environments, and that such an environment may be in any of a set E of states. In order to represent the effect that an agent's actions have on an environment, we introduce a *state transformer* function, τ (cf. [4, p154]). The idea is that τ takes as input an environment state $e \in E$ and an action $\alpha \in Ac$, and returns the environment state that would result from performing α in

e. Thus $\tau : E \times Ac \rightarrow E$. Note that we are implicitly assuming that environments are *deterministic*: there is no uncertainty about the result of performing an action in some state [11, p46]. In addition, we assume that the only way an environment state can change is through the performance of an action on the part of an agent (i.e., the environment is *static* [11, p46]). Dropping these assumptions is not particularly problematic and does not alter any of our results, although it does make the formalism somewhat more convoluted. We leave the reader to make the required modifications. Formally, we define an environment *Env* to be a triple (E, τ, e_0), where E is a set of environment states as above, τ is a state transformer function, and $e_0 \in E$ is the initial state of *Env*.

A *run* of an agent/environment system can be thought of as an infinite sequence:

$$r : g_0 \xrightarrow{c_0} g_1 \xrightarrow{c_1} g_2 \xrightarrow{c_2} g_3 \xrightarrow{c_3} \cdots \xrightarrow{c_{u-1}} g_u \xrightarrow{c_u} \cdots$$

In such a run, g_0 is the initial state of the system (comprised of the initial state of the environment and the initial state of the agent) and $c_0 \in C$ is the *choice* dictated by the agent's meta-level control strategy on the basis of it's initial state. The state $g_1 = (e_1, l_1)$ is that which results after the agent has made its choice c_0. If the agent chose to *act* (that is, if $c_0 = a$), then $e_1 = \tau(e_0, \mathcal{A}(l_0))$ and $l_1 = (\mathcal{N}(e_0, b_{l_0}), i_{l_0})$, that is, the environment state e_1 is that which results from the agent performing its chosen action in the initial state, and the internal state l_1 is that which results from the agent updating its beliefs via its belief revision function and not changing its intentions (since it did not deliberate).

If, however, the agent chose to *deliberate* at time 0 (i.e., if $c_0 = d$) then $e_1 = e_0$ (i.e., the environment remains unchanged, since the agent did not act), and $l_1 = (\mathcal{N}(e_0, b_{l_0}), \mathcal{D}(l_0))$ (i.e., the agent's beliefs are updated as in the previous case, and the agent's intentions are updated through its deliberation function $\mathcal{D}$.

Formally, an infinite sequence $(g_0, g_1, g_2, \ldots)$ over G represents a run of an agent $Ag = (\mathcal{M}, \mathcal{D}, \mathcal{A}, \mathcal{N}, l_0)$ in an environment $Env = (E, \tau, e_0)$ iff $g_0 = (e_0, l_0)$ and $\forall u \in \mathbb{N}$, we have

$$g_{u+1} = \begin{cases} (e_u, (\mathcal{N}(e_u, b_{l_u}), \mathcal{D}(l_u))) & \text{if } \mathcal{M}(l_u) = d \\ (\tau(e_u, \mathcal{A}(i_{l_u})), (\mathcal{N}(e_u, b_{l_u}), i_{l_u})) & \text{if } \mathcal{M}(l_u) = a. \end{cases}$$

We will denote by $r(Ag, Env)$ the run of agent Ag in environment *Env*, and let *Run* be the set of all possible runs.

2.3 Optimal Behaviour

In order to express the *value*, or *utility* of a run, we introduce a function $V : Run \rightarrow \mathbb{R}$, which assigns real numbers indicating "payoffs" to runs. Thus V essentially captures a standard decision-theoretic notion of utility. A *task environment* is defined to be a pair (Env, V), where *Env* is an environment, and $V : Run \rightarrow \mathbb{R}$ is a utility function. We say an agent Ag is *optimal* with respect to a task environment (Env, V) if there is no agent Ag' such that $V(r(Ag', Env)) > V(r(Ag, Env))$. Again, this is in essence the by-now standard notion of an optimal agent (cf. [12, p583]).

Ultimately, an agent is simply an elaborate action selection function. The *components* of an agent — its meta-level control strategy, deliberation, action, and next-state function — are there *in the service* of this decision making. An obvious question is therefore whether or not we can define what it means for such a component to be optimal. Let us consider the case of the meta-level control. Suppose that in some situation, the meta-level control strategy chose to deliberate rather than act, and as a consequence, lost some utility. (Imagine that the agent was about to be hit by a speeding car, and instead of choosing to jump, chose to deliberate about which way to jump.) Then clearly, the meta-level control strategy was sub-optimal in this case; it would have been better to have chosen differently. This leads us to the following definition: a meta-level control strategy $\mathcal{M}$ is *sub-optimal* if there is some other meta-level control strategy $\mathcal{M}'$ such that if the agent used $\mathcal{M}'$ instead of $\mathcal{M}$, it would obtain a higher utility. Formally, if $(\mathcal{M}, \mathcal{D}, \mathcal{A}, \mathcal{N}, l_0)$ is an agent, then $\mathcal{M}$ if optimal (with respect to (Env, V), $\mathcal{D}$, $\mathcal{A}$, and $\mathcal{N}$)

if there is no $\mathcal{M}'$ such that $V(r(\mathcal{M}', \mathcal{D}, \mathcal{A}, \mathcal{N}, l_0), Env) > V(r(\mathcal{M}, \mathcal{D}, \mathcal{A}, \mathcal{N}, l_0), Env)$. In a similar way, we can define optimality for $\mathcal{D}$, $\mathcal{A}$, and $\mathcal{N}$ — the details are left to the reader. Notice that optimality of a component is defined not only with respect to a task environment, but also with respect to the other components of an agent. The following theorem captures the relationship between optimality for an agent and the optimality of its components.

Theorem 1 *If an agent is optimal with respect to some task environment, then the components of that agent are mutually optimal.*

Proof: Suppose that $Ag = (\mathcal{M}, \mathcal{D}, \mathcal{A}, \mathcal{N}, l_0)$ is globally optimal with respect to (Env, V), but that one component is sub-optimal. Assume this component is $\mathcal{M}$ (the cases for $\mathcal{D}$, $\mathcal{A}$, or $\mathcal{N}$ are identical). Then $V(r(\mathcal{M}', \mathcal{D}, \mathcal{A}, \mathcal{N}, l_0), Env) > V(r(\mathcal{M}, \mathcal{D}, \mathcal{A}, \mathcal{N}, l_0), Env)$ for some $\mathcal{M}'$ such that $\mathcal{M}' \neq \mathcal{M}$. But in this case, Ag is not optimal with respect to (Env, V), which is a contradiction. □

Notice that the implication in this theorem cannot be strengthened to a biconditional: the fact that the components of an agent are mutually optimal does imply that the agent is itself optimal. We can think of agents that have mutually optimal components but that are globally sub-optimal as having achieved a kind of local maxima: an optimality of sorts, but not the best that could be achieved.

For the remainder of this paper, we will be particularly concerned with the relationship between just two of the components of an agent: its meta-level control strategy and deliberation component. We shall therefore assume from here on that an agent's next-state function and action function are fixed and optimal.

3 Real-Time Task Environments

It should be clear that the performance of an agent is very much dependent on the nature of the task environment in which it is situated. An agent that performs badly in one task environment may do well in one that has different properties. An understanding of the relationship between agents and the task environments they occupy is therefore likely to be of benefit when we come to actually building agents that will operate in real environments. For this reason, we now turn our attention to these relationships. In particular, we consider how various environmental properties can correspond to properties of agents and their components. Although typologies of environment properties have appeared in the literature (e.g., [11, p46]), the most important single environment property is that of being *real-time*. A real-time task environment is simply one in which time is significant [12, p585]. In a real-time task environment, an agent cannot afford to deliberate indefinitely — it must make decisions in time for these decisions to be useful (cf. the notion of reactivity in [16]). Real-time task environments are problematic simply because if an agent is to operate successfully in such an environment, then it must achieve the successful trade-off between deliberation and action that we discussed above.

How are we to define what it means for a task-environment to be real-time? Intuitively, a real-time task environment is one in which an agent is penalised for any wasted effort. How might an agent waste effort? There are essentially two possibilities. First, an agent is wasting effort if it is expending resources attempting to achieve the "wrong" intentions. To see what we mean by this, consider the Tileworld scenario, introduced by Pollack and Ringuette [8], and used by Kinny and Georgeff in their investigation into agent commitment strategies [6]. In this environment, an agent is attempting to shove blocks into various holes that appear in a two-dimensional grid-world. Unfortunately, the holes themselves arbitrarily appear and disappear. Now if an agent is attempting to achieve an intention to shove a block into a particular hole even when that hole has vanished, then it is wasting effort — it would intuitively do better to reconsider its intentions. A similar waste of effort occurs if an agent fails to exploit a serendipitous situation (for example when a hole appears to the side of an agent as it pushes a block). A second type of wasted effort occurs if an agent has "correct" intentions, but is not acting on them — in such a situation, an agent is engaging in unnecessary deliberation.

Situation number	Optimal intentions?	Chose to deliberate?	Changed focus?	$\mathcal{M}$ optimal?	$\mathcal{D}$ optimal?
1	No	No	—	No	—
2	No	Yes	No	Yes	No
3	No	Yes	Yes	Yes	Yes
4	Yes	No	—	Yes	—
5	Yes	Yes	No	No	Yes
6	Yes	Yes	Yes	No	No

Table 1: Practical Reasoning Situations

In order to formally define what we mean by a real-time task environment, we need to define what it means for an agent to have *optimal intentions*. Intuitively, an agent has optimal intentions if there is no good reason for changing them — if, given the information available to the agent, an optimal deliberation function would not choose to change them. Formally, if $(\mathcal{M}, \mathcal{D}, \mathcal{A}, \mathcal{N}, l_0)$ is an agent that is currently in state (b, i), and that is situated in task environment (Env, V), the its intention set i is optimal if $\mathcal{D}'((b, i)) = i$ for a deliberation strategy $\mathcal{D}'$ that is optimal for $(\mathcal{M}, \mathcal{D}, \mathcal{A}, \mathcal{N}, l_0)$.

Given this, we say a task environment (Env, V) is real-time iff for any optimal run $(g_0, g_1, \ldots)$ of an agent $(\mathcal{M}, \mathcal{D}, \mathcal{A}, \mathcal{N}, l_0)$ in this task environment, there is no $u \in \mathbb{N}$ such that either i_{l_u} is optimal and $c_u = d$ or else i_{l_u} is not optimal and $c_u = a$.

The possible interactions between meta-level control and deliberation in real-time task environments are summarised in Table 1 (adapted and extended from [2, p353]). Consider situation (1). In this situation, the agent does not have optimal intentions, and would hence do well to deliberate. However, it does not choose to deliberate and hence the meta-level reasoning strategy that chose to act was sub-optimal. In situation (2), the agent again has sub-optimal intentions, but this time chooses to deliberate, rather than act. Unfortunately, the agent's deliberation function $\mathcal{D}$ does not change focus, and is hence sub-optimal. Situation (3) is essentially the same as situation (2), with the exception that this time, the deliberation function *does* change focus, and hence both the meta-level reasoning and deliberation strategy of the agent are behaving optimally. In situation (4), the agent has optimal intentions, and does not choose to deliberate. Since the intentions are optimal, the meta-level control strategy is obviously correct not to deliberate in this situation, and is hence optimal. In situation (5), the agent has optimal intentions, but this time chooses to deliberate; the deliberation strategy, however, does not change focus. Hence while the meta-level control strategy is clearly sub-optimal, the deliberation strategy is optimal. Situation (6) is as situation (5), except that this time, the deliberation function changes focus. In this case, both the meta-level control and deliberation components must be sub-optimal, since the agent wasted time deliberating, and then modified its intentions despite the fact that there is no reason to do so.

From the discussion above, we can extract the following simple principle: in real-time environments, a meta-level control strategy should choose to deliberate rather than act *only* when an optimal deliberation strategy would change focus. We will say a meta-level control strategy $\mathcal{M}$ is *sound* with respect to an optimal deliberation strategy $\mathcal{D}$ iff whenever $\mathcal{M}$ chooses to deliberate, $\mathcal{D}$ chooses to change focus (i.e., if $M(l) = d$ implies $\mathcal{D}(l) \neq i_l$). Similarly, we say $\mathcal{M}$ is *complete* with respect to $\mathcal{D}$ iff whenever $\mathcal{D}$ *would* change focus, $\mathcal{M}$ chooses to deliberate (i.e., if $\mathcal{D}(l) \neq i_l$ implies $M(l) = d$).

Given this situation, one might wonder what is the point of having both meta-level control *and* deliberation components, as an optimal meta-level control strategy need only run the deliberation function as a subroutine to see if it would change focus, and choose to deliberate just in case the deliberation strategy *does* change focus. This would indeed be a successful strategy if the cost of running the meta-level control strategy was roughly equal to the cost of deliberation. However, as we pointed out earlier, we require that the cost of meta-level control be *significantly less* than that of deliberation. Under this

assumption, running the deliberation component in order to decide whether to deliberate is not a realistic option.

We can easily establish the following theorem, which relates sound and complete meta-level control strategies to real-time environments.

Theorem 2 *For real-time task environments, a meta-level control-strategy is optimal with respect to an optimal deliberation strategy iff it is sound and complete with respect to this deliberation strategy.*

Proof: For the left to right implication, assume $\mathcal{M}$ is optimal. Then we need to show that $\mathcal{M}$ is sound and complete. For soundness, assume that $\mathcal{M}(l) = d$ (the meta-level control strategy says deliberate) but that $\mathcal{D}(l) = i_l$ (the deliberation function does not choose to change focus). Then since this is a real-time task environment, $\mathcal{M}$ is not optimal. This is a contradiction, so if $\mathcal{M}$ is optimal, it is sound. For completeness, assume $\mathcal{D}(l) \neq i_l$ but that $\mathcal{M}(l) = a$. Then since the task environment is real-time, $\mathcal{M}$ is not optimal. This is a contradiction, so if $\mathcal{M}$ is optimal, it is complete. For the right to left implication, assume $\mathcal{M}$ is sound and complete with respect to $\mathcal{D}$, but that it is not optimal. If it is not optimal, then it must be making a wrong decision at some point, i.e., it must be choosing to deliberate when it would be better to act, or else to act when it would be better to deliberate. Consider the first case, where $\mathcal{M}(l) = d$. Since $\mathcal{M}$ is sound, we know that $\mathcal{D}(l) \neq i_l$. Since $\mathcal{D}$ is optimal, the agent's intentions in l must be sub-optimal, hence acting on them would be sub-optimal. Hence deliberation must be the optimal choice. For the second case, suppose $\mathcal{M}(l) = a$. As $\mathcal{M}$ is sound and complete, we know that $\mathcal{M}(l) = d$ iff $\mathcal{D}(l) \neq i_l$. Since $\mathcal{M}(l) = a$, this means that $\mathcal{D}(l) = i_l$, hence the agent's intentions are optimal, and since the environment is real-time, deliberating on them would be sub-optimal. Hence acting is the optimal choice, so $\mathcal{M}$ is optimal. □

In this same way, we say a deliberation strategy $\mathcal{D}$ is sound (with respect to optimal meta-level control strategy $\mathcal{M}$) iff it changes focus when the meta-level control strategy chooses to deliberate, (i.e., if $\mathcal{D}(l) \neq i_l$ implies $M(l) = d$), and *complete* iff whenever the meta-level control strategy chooses to deliberate, it changes focus (i.e., $M(l) = d$ implies $\mathcal{D}(l) \neq i_l$). Not surprisingly, the following theorem can be established using the same techniques as Theorem 2.

Theorem 3 *For real-time task environments, a deliberation strategy is optimal with respect to an optimal meta-level control strategy iff it is sound and complete with respect to this meta-level control strategy.*

4 An Example

In the previous section, we discussed the notion of a real-time task environment, and investigated the relationship between meta-level control and deliberation in such task environments. In this section, we show how four illustrative practical reasoning scenarios (introduced in [2]) can be represented within our framework. (More accurately, Bratman and colleagues give six scenarios, since there are two variants each of scenarios one and four. However, as we discuss below, these variants are meaningless in our framework.)

4.1 Scenario One

All four scenarios are based on the following basic story: Rosie is an agent that has been assigned the task of repairing a malfunctioning VDU. As a result of some task analysis, she has decided that this might best be done by replacing the CRT (which she believes is burnt out), and so she has adopted the intentions of going to the VDU armed with a replacement CRT, and then using this new tube to fix the VDU. In the first scenario, Rosie arrives at the VDU to find that the CRT is not burnt out: the contrast has just been turned way down. She therefore has the option of fixing the VDU by adjusting the contrast.

Beliefs	
w	VDU working
c	CRT burnt out
d	Contrast turned down
b_1	Adjust contrast is better
r	CRT can be fixed by re-wiring
b_2	Re-wiring is better
s	Spare VDU
b_3	Spare VDU is better

Intentions	
i_o	Fix VDU using original CRT
i_c	Fix VDU by adjusting contrast
i_r	Fix VDU by re-wiring
i_a	Fix VDU by using alternative CRT
i_v	Go to VDU

Table 2: Rosie's Possible Beliefs and Intentions

This information is sufficient for her meta-level control strategy to decide that it is worth deliberating, and in so doing, Rosie finds that adjusting the contrast is cheaper than replacing the CRT. She this adopts the new intention of adjusting the contrast. She then acts, adjusting the contrast and completes her initial task.

In this, and all other scenarios, we represent Rosie's world as a set of propositions. The propositions of interest to us are summarised in Table 2. While the intended interpretation for most of these is self-evident, some require additional explanation: s is intended to capture the presence of the additional CRT in scenarios three and four; b_1 is intended to capture the fact that Rosie knows that if it is possible to fix the VDU by just adjusting the contrast then this is a better option than using the CRT she carries with her; b_2 is intended to capture the fact that rewiring the faulty CRT is the best option, and b_3 is intended to capture the fact that the additional CRT in scenarios three and four is superior to the CRT she carries with her.

In addition, we will also represent Rosie's possible intentions as propositions: see Table 2. Again, most of these are self-explanatory, but i_v is needed to capture Rosie's initial progress from wherever she picks up the first CRT to wherever the broken VDU is. For simplicity we will assume that each of these intentions can be achieved by a single action (though each of these could equally well be a series of actions). Thus the action to achieve intention i_r is α_r, the action to achieve intention i_v is α_v, and so on.

We can now formalise Rosie's reasoning. Initially the state of the world is $e_0 = \{\neg w, \neg c, d\}$ (the VDU is not working, the CRT is not burnt out, and the contrast is turned down). Rosie's initial internal state l_0 is thus: $(\{\neg w, c, \neg d, b_1\}, \{i_v, i_o\})$. She thus begins scenario one with false beliefs, since she wrongly believes that the CRT is burned out. Note that Rosie's beliefs also include the preference information b_1. She initially has two intentions: to fix the VDU using the original CRT, and to go to the VDU.

The first part of Rosie's operation is to decide whether to deliberate or act. She chooses to act, and executes the action α_v that achieves her intention i_v, and thus arrives at the VDU. At this point she deliberates, and removes the now-achieved intention of moving to the VDU from her intention set, so that the previously adopted intention of fixing the VDU using the CRT she brought with her becomes the main focus. At this point she can identify the real state of the world, and her next-state function $\mathcal{N}$ updates her beliefs to reflect this. Her internal state becomes: $l_1 = (\{\neg w, \neg c, d, b_1\}, \{i_o\})$. The state of the external world is unchanged: $e_1 = e_0$.

Rosie again applies her meta-level control strategy:

$$\mathcal{M}(l) = \begin{cases} d & \text{if } \{\neg c, d, b_1\} \subseteq b_l \text{ or } \{\neg c, r, b_2\} \subseteq b_l \text{ or } \{c, s, b_3\} \subseteq b_l \\ a & \text{otherwise.} \end{cases}$$

Thus there are three situations in which she will choose to deliberate, all of which can be glossed as "there is now some reason to suspect that there is a better alternative to repair the VDU". Clearly this is just an illustrative fragment of the complete meta-level control function which is appropriate to this example. Since Rosie now believes $\neg c$, she chooses to deliberate. That is, $\mathcal{M}(l_1) = d$ since the CRT is known to not be burnt out, the contrast is known to be turned down, and it is known that adjusting the contrast gives a better means of fixing the VDU than replacing the CRT. To find the result of deliberation, we need to define $\mathcal{D}$. We have:

$$\mathcal{D}(l) = \begin{cases} \{i_c\} & \text{if } \{\neg c, d, b_1\} \subseteq b_l \\ \{i_r\} & \text{if } \{\neg c, r, b_2\} \subseteq b_l \\ \{i_a\} & \text{if } \{c, s, b_3\} \subseteq b_l \\ l_i & \text{otherwise.} \end{cases}$$

The deliberation function $\mathcal{D}$ thus decides to adjust the contrast: $\mathcal{D}(l_1) = \{i_c\}$. Note that $\mathcal{D}$ should really check that the agent has a means of adopting the intention before it decides to adopt it — if Rosie is unable to adjust the contrast (because she has the wrong kind of gripper for instance) then however good a solution this might be, there is no point in changing focus to try and achieve it. For our purposes, we can ignore this subtlety, however.

After deliberation, Rosie's internal state becomes: $l_2 = (\{\neg w, \neg c, d, b_1\}, \{i_c\})$, while the external world remains unchanged: $e_2 = e_1 = e_0$. This time $\mathcal{M}$ chooses to act, and since $\mathcal{A}(l_2) = \alpha_c$, the contrast is adjusted, which repairs the VDU. This change in the world causes Rosie to revise her beliefs about the state of the VDU and the contrast control. The final state of the environment is thus $e_3 = \{w, \neg c, \neg d\}$, while Rosie's internal state is $l_3 = (\{w, \neg c, \neg d, b_1\}, \emptyset)$.

The complete run for scenario one is thus:

$$r_1 : g_0 \xrightarrow{a_v} g_1 \xrightarrow{d} g_2 \xrightarrow{a_c} g_3$$

4.2 Scenario Two

In this scenario, Rosie arrives at the VDU to find that the CRT is not burnt out and can be fixed by re-wiring. However, this fix will only be short term, and the CRT will soon burn out anyway. This information is sufficient for Rosie's meta-level control strategy to decide it is not worth deliberating to see if she is able to fix the VDU by rewiring, and so she acts, replacing the CRT in line with her unchanged intention. The start this scenario is described by:

$$\begin{aligned} e_0 &= \{\neg w, \neg c, r\} \\ l_0 &= (\{\neg w, c, \neg r, \neg b_2\}, \{i_v\}) \end{aligned}$$

So, although the CRT is not burnt out and the VDU can be fixed by re-wiring (facts that Rosie initially does not know), Rosie *does* know that re-wiring is a worse option than replacing the CRT. After moving to the VDU, popping the intention stack, and revising beliefs, just as in the previous scenario, the environment state remains unchanged but Rosie's internal sate is $l_1 = (\{\neg w, \neg c, r, \neg b_2\}, \{i_o\})$.

Rosie then applies her meta-level control strategy, and despite the fact that there is reason for her to suspect that deliberation might lead to an alternative means of repairing the VDU (a situation which is actually true), $\mathcal{M}$ returns a because Rosie also knows that fixing the CRT by re-wiring is a worse option than the one she has already. Thus she can reject the idea of changing her focus without going as far as establishing whether or not she can build a new plan in order to fix the VDU. Having decided to act, Rosie performs $\mathcal{A}(i_o) = \alpha_o$ and the situation becomes:

$$e_2 = \{w, \neg c, r\}$$
$$l_2 = (\{w, c, r, \neg b_2\}, \emptyset)$$

The complete run for Scenario Two is thus:

$$r_2 : g_0 \xrightarrow{a_v} g_1 \xrightarrow{a_o} g_2$$

4.3 Scenario Three

In Scenario Three, Rosie arrives at the VDU to find a spare (and therefore free) CRT sitting by the terminal, but notes that the spare is inferior to the tube she brought with her. Her meta-level control mechanism therefore realises that there is no advantage to seeing if the new tube can be used, and so chooses to act. Rosie then replaces the CRT in line with her original intention. Scenario Three thus begins with the following state of affairs:

$$e_0 = \{\neg w, c, s\}$$
$$l_0 = (\{\neg w, c, \neg s, \neg b_3\}, \{i_v\})$$

As before, Rosie proceeds to the VDU and this time finds the spare tube. After belief revision, the environment state remains unchanged but Rosie's internal state becomes $l_1 = (\{\neg w, c, s, \neg b_3\}, i_0)$. This time $\mathcal{M}$ tells her to act, because the newly visible CRT is worse than the one she is carrying with her. She acts, $\mathcal{A}(l_1) = \alpha_o$ by replacing the CRT and the situation becomes:

$$e_2 = \{\neg w, \neg c, s\}$$
$$l_2 = (\{\neg w, \neg c, s, \neg b_3\}, \emptyset)$$

The complete run for Scenario Three is thus:

$$r_3 : g_0 \xrightarrow{a_v} g_1 \xrightarrow{a_o} g_2$$

4.4 Scenario Four

In Scenario Four, Rosie arrives at the VDU to again find a spare CRT sitting by the terminal, and this time notes that the spare is superior to the tube she brought with her. Her meta-level control mechanism therefore realises that there is considerable advantage to seeing if the new tube can be used since the saving in the cost of the tube is greater than the cost of deliberation. So she chooses to deliberate. Deliberation results in the adoption of the intention to use the new tube, and Rosie then replaces the CRT in line with this new intention. This scenario is almost the same as the third, except that this time the "new" CRT is superior to the one that Rosie brings with her. Thus the initial situation is:

$$e_0 = \{\neg w, c, s\}$$
$$l_0 = (\{\neg w, c, \neg s, b_3\}, \{i_v\})$$

After moving to the VDU and revising beliefs, the environment is unchanged ($e_1 = e_0$) but Rosie's internal state is $l_1 = (\{\neg w, c, s, b_3\}, \{i_0\})$. This time $\mathcal{M}(l_1) = d$ and $\mathcal{D}(l_1) = \{i_a\}$. After this, the environment state again remains unchanged but Rosie's internal state is $l_2 = (\{\neg w, c, s, b_3\}, \{i_a\})$, and Rosie proceeds to act $\mathcal{A}(l_2) = \alpha_a$ giving the following global state:

$$e_3 = \{\neg w, \neg c, s\}$$
$$l_3 = (\{\neg w, \neg c, s, b_3\}, \emptyset)$$

The complete run for Scenario Four is thus:

$$r_1 : g_0 \xrightarrow{a_v} g_1 \xrightarrow{d} g_2 \xrightarrow{a_a} g_3$$

There are several points to note about this example. The first is that both $\mathcal{M}$ and $\mathcal{D}$ are optimal for the cases given. There is no set of actions which could be chosen to give a better result. The second is that it is easy to alter the example so that Rosie is not optimal. Consider what would happen in Scenario Four if she had no means of using the additional CRT (which would mean that there was no intention i_a, or, worse no action α_a for achieving i_a). $\mathcal{M}$ would choose to deliberate since the CRT is superior, but either this deliberation would not change the intentions (if there was no i_a), or when Rosie came to act on the changed intention, she would be unable to achieve that intention and would have to revert to i_o. The final point to note is that it is this consideration of intentions and actions which justifies our assumption that the time cost of $\mathcal{M}$ is less than that of $\mathcal{D}$. Deliberation will typically involve an expensive activity such as building and evaluating the quality of plans to achieve some set of alternative intentions. Although that activity might be as simple as looking to see if there is some alternative intention which can be adopted, as here, it is still an overhead.

5 Generalised Meta-Level Reasoning

In this section, we will sketch out how an agent might use *higher-order* meta-level control strategies in its architecture, and what role such strategies might play. What do we mean by a higher-order meta-level control strategy? Let us refer to the meta-level control strategies as described above as *first-order* meta-level strategies. Such strategies merely choose whether to deliberate or to act. A *second-order* meta-level control strategy can be thought of as *selecting* which first-order meta-level control strategy to use. For example, a second-order meta-level control strategy might examine the agent's beliefs to see how dynamic the agent's environment is. If it determines that the environment is highly dynamic (i.e., the rate of world change is high [6]), then it might select a cautious first-order meta-level control strategy — one which frequently causes the agent to deliberate. If, in contrast, the environment is relatively static (the rate of world change is low), then it might select a *bold* meta-level control strategy (one that favours action over deliberation).

It is easy to imagine an agent with a "tower" of such meta-level control strategies, with nth-order strategy selecting which strategy to use at level $n-1$. The idea is very similar to the use of meta-language hierarchies in meta-logic [7, 15].

We can incorporate such higher-order meta-level reasoning into our formal model with ease. First, let $MLC_1 = L \rightarrow C$ be the set of all *first-order* meta-level control strategies. These are the meta-level control strategies that we discussed above. Then define $MLC_u = L \rightarrow MLC_{u-1}$, for all $u \in \mathbb{N}$ such that $u > 1$. Thus MLC_2 is the set of all second-order meta-level control strategies, MLC_3 is the set of all third-order meta-level control strategies, and so on. An agent becomes a 5-tuple, $(\mathcal{M}_n, \mathcal{D}, \mathcal{A}, \mathcal{N}, l_0)$, where $\mathcal{M}_n$ is an nth order meta-level control strategy and the agent's other components are as before. Given this, we can redefine what it means for a run to represent a history of an agent in an environment. Formally, an infinite sequence $(g_0, g_1, g_2, \ldots)$ over G represents a run of an agent $Ag = (\mathcal{M}_n, \mathcal{D}, \mathcal{A}, \mathcal{N}, l_0)$ in an environment $Env = (E, \tau, e_0)$ iff $g_0 = (e_0, l_0)$ and $\forall u \in \mathbb{N}$, we have

$$g_{u+1} = \begin{cases} (e_u, (\mathcal{N}(e_u, b_{l_u}), \mathcal{D}(l_u))) & \text{if } \mathcal{M}_n(l_u)\overbrace{(l_u)\cdots(l_u)}^{n-1 \text{ times}} = d \\ (\tau(e_u, \mathcal{A}(i_{l_u})), (\mathcal{N}(e_u, b_{l_u}), i_{l_u})) & \text{if } \mathcal{M}_n(l_u)\underbrace{(l_u)\cdots(l_u)}_{n-1 \text{ times}} = a. \end{cases}$$

Notice that agents which make use of higher-order meta-level control are strictly speaking no more powerful than "ordinary" agents, as defined earlier. For every higher-order agent there is an "ordinary" agent that behaves in exactly the same way. The point is that from the point of view of an agent designer, it may make sense to divide the functionality of the agent up into different levels of meta-reasoning.

6 Conclusions

In this paper, we have investigated the relationship between the deliberation, action, and meta-level control components of a practical reasoning architecture. While this relationship has previously been investigated from an experimental perspective (particularly by Kinny [6]), we have in contrast attempted a mathematical analysis. We have demonstrated how it is possible to construct a simple but, we argue, realistic model of practical reasoning agents of the type investigated by Kinny and Georgeff, and we have established some basic properties of such agents when placed in different types of task environment. We have focussed in particular on real-time task environments, since these are, we believe, the most common class of real-world task environment that one encounters.

This work was originally instigated in an attempt to relate the work of Russell and Subramanian on bounded-optimal agents (agents that perform as well as any agent can do under certain architectural constraints [12]) to the increasingly large literature on BDI agents [5, 2, 9, 10]. While this initial investigate led us into some areas we had not initially anticipated visiting, we believe that investigating the implications of bounded-optimal agents for BDI model will be an interesting research issue, and one that we hope to investigate in future work. Another issue that we hope to consider is the moving from individual agents to multi-agent systems.

References

[1] M. E. Bratman. *Intentions, Plans, and Practical Reason.* Harvard University Press: Cambridge, MA, 1987.

[2] M. E. Bratman, D. J. Israel, and M. E. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4:349–355, 1988.

[3] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.

[4] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge.* The MIT Press: Cambridge, MA, 1995.

[5] M. P. Georgeff and A. L. Lansky. Reactive reasoning and planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pages 677–682, Seattle, WA, 1987.

[6] D. Kinny and M. Georgeff. Commitment and effectiveness of situated agents. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 82–88, Sydney, Australia, 1991.

[7] D. Perlis. Meta in logic. In P. Maes and D. Nardi, editors, *Meta-Level Architectures and Reflection*, pages 37–49. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1988.

[8] M. E. Pollack and M. Ringuette. Introducing the Tileworld: Experimentally evaluating agent architectures. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pages 183–189, Boston, MA, 1990.

[9] A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In R. Fikes and E. Sandewall, editors, *Proceedings of Knowledge Representation and Reasoning (KR&R-91)*, pages 473–484. Morgan Kaufmann Publishers: San Mateo, CA, April 1991.

[10] A. S. Rao and M. P. Georgeff. An abstract architecture for rational agents. In C. Rich, W. Swartout, and B. Nebel, editors, *Proceedings of Knowledge Representation and Reasoning (KR&R-92)*, pages 439–449, 1992.

[11] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice-Hall, 1995.

[12] S. Russell and D. Subramanian. Provably bounded-optimal agents. *Journal of AI Research*, 2:575–609, 1995.

[13] S. J. Russell and E. Wefald. *Do the Right Thing — Studies in Limited Rationality.* The MIT Press: Cambridge, MA, 1991.

[14] H. A. Simon. *The Sciences of the Artificial (second edition).* The MIT Press: Cambridge, MA, 1981.

[15] R. Turner. *Truth and Modality for Knowledge Representation.* Pitman Publishing: London, 1990.

[16] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.