Translating a Modal Language with Embedded Implication into Horn Clause Logic

Matteo Baldoni, Laura Giordano and Alberto Martelli

Dipartimento di Informatica - Università di Torino
C.so Svizzera 185 - 10149 TORINO
E-mail: {baldoni,laura,mrt}@di.unito.it
Phone: +39 11 74 29 111 Fax: +39 11 75 16 03

Abstract. In this paper we present a method for translating Horn clauses extended with modalities and embedded implication (which provide reasoning capabilities in a multiagent situation and hypothetical reasoning) into Horn clauses, therefore suitable for SLD resolution. The translation takes two steps: the first one eliminates embedded implications by introducing new modalities; the second eliminates modalities by adding an argument which represents the worlds of the Kripke semantics to all predicates.
Keywords: Multimodal Logic, Embedded Implication, Translation.

1 Introduction

Modal logics are widely used in computer science and artificial intelligence to deal with knowledge and beliefs, time, actions, and several researchers have proposed modal extensions of logic programming languages [7, 1, 13, 19, 17, 6]. In particular, in [5] we have defined a modal logic programming language which allows both multiple modalities and embedded implications. The language has been shown to be well suited for structuring knowledge, and, in particular, for defining module constructs within programs, for representing agent beliefs and also for hypothetical reasoning. This language has a goal directed operational semantics which has been proved to be sound and complete with respect to the Kripke semantics. The operational derivability of a goal is defined with respect to a notion of *modal context*, which consists of a sequence of modal operators. The modal context keeps track of the new clauses which are added to the program when evaluating implication goals.

The modal operators of this language are rather specific, since the language was defined for providing constructs for modularizing programs, and for allowing simple reasoning capabilities in a multiple agent environment. However, the goal directed procedure can be extended to similar languages whose modal operators have different properties.

In the next section we introduce the modal language, and recall its operational semantics. The goal directed procedure gives a precise definition of the operation behaviour of a program, and provides a means for executing a program. However the actual implementation of the procedure can raise several problems. The simplest solution of building an interpreter (say in Prolog), may turn out to be extremely inefficient, since the interpreter will have to deal with the modal context, and, in particular, with the dynamically added clauses.

In this paper we present a different approach, based on translating our language into Horn clause logic, so that the translated program can be executed by any Prolog interpreter or compiler, with the advantage that many features, such as unification or variable renaming, are directly provided. Furthermore, a

real program usually needs to use built-in predicates and extra logical features, which, again, are provided by the Prolog environment (as, for instance, *cut*).

A possible disadvantage of translation methods is that the translated program can be very different from the original one, so that the steps of the proof will be entirely unrelated with the original proof. We will show that this is not the case with our translation, and that there is a precise correspondence between the two kinds of proofs.

The translation method consists of two steps, which are presented in Section 3 and 4. In the first step all embedded implications are eliminated so as to obtain a program consisting only of modal Horn clauses. This step requires the introduction of new modal operators which ensure that the extracted clauses can be used only in the right way. In the second step modalities are eliminated by adding to all predicates an argument which represents the modal context.

Section 5 describes some possible extensions of this work, and compares the approach of this paper with translation methods which have been developed in the context of modal theorem proving.

2 The source language

In this section we show the source programming language $\mathcal{L}$ introduced in [5]. It extends Horn clause language with k modal operators $[a_1], \ldots, [a_k]$, where the a_i's are constants, and a modal operator $\Box$; moreover, it allows implication goals to occur in goals and in clause bodies. $\mathcal{L}$ is an extension of the language defined in [13, 3] and more details can be found in [4]. It provides a simple way to formulate reasoning capabilities in multiple agent situation; intuitively $[a_i]\alpha$ can be read "agent a_i believes α" and $\Box\alpha$ as "α is common knowledge of agents"; furthermore it allows both to define module construct and to perform hypothetical reasoning.

2.1 Syntax of $\mathcal{L}$

Let A be an atomic formula of the form $p(t_1, \ldots, t_s)$, with p a predicative symbol and $t_1, \ldots, t_s$ terms of the language, and T a distinguished symbol (*true*). The syntax of the language $\mathcal{L}$ is the following

$$\begin{aligned} G &::= T \mid A \mid G_1 \ , \ G_2 \mid LG \mid L(D \supset G) \\ D &::= H \mid H \mathrel{:-} G \mid D_1 \ . \ D_2 \mid LD \\ H &::= A \mid LH \\ L &::= [a_i] \mid \Box \end{aligned}$$

where G stands for a *extended goal*, D for a *extended clause*, H for a *extended clause head* or a *fact* (we will refer to extended clauses and goals without the word extended, when no confusion arises). Note that all free variables in a clause are (implicitly) assumed to be universally quantified. For example, the clause

$$p(X) \mathrel{:-} \Box((r(a) \ . \ q(X) \mathrel{:-} r(X)) \supset q(X)) \tag{1}$$

is implicitly universally quantified in the following way

$$(\forall X)(p(X) \mathrel{:-} \Box((r(a) \ . \ (\forall X)(q(X) \mathrel{:-} r(X))) \supset q(X)))$$

and so, the variable X in the nested clause $(\forall X)(q(X) \mathrel{:-} r(X)))$ is different from the X in $p(X)$; in other words, (1) is equivalent to

$$p(X) \mathrel{:-} \Box((r(a) \ . \ q(Y) \mathrel{:-} r(Y)) \supset q(X)),$$

where Y is any variable different from X. Moreover, in the following, D will be interchangeably regarded as a conjunction or a set of universally quantified closed clauses. A program P consists of a set of clauses D.

In this language, embedded implications are allowed and modal operators $[a_i]$ and $\Box$ can freely occur in front of clauses, in front of clause heads, in front of each goal and, in particular, in front of embedded implications (or implication goals). $[a_i]G$ is a goal and means that G has to be proved in the beliefs of agent a_i, while $[a_i]D$ means that the clause D is part of the beliefs of agent a_i. As regards implication goals, they *must* be preceded by at least one modal operator, for example $[a_i](D \supset G)$[1].

Note that, for uniformity with Prolog notation, we have used the symbol ":–" as the implication in clauses, while we have used the simbol "$\supset$" in embedded implications. However, we regard both of the symbols as material implication. Moreover, we have used the symbol "," and "." as the conjunction in goals and clauses, respectively.

To give an idea of how a program in this language is defined, let us consider the following example. It is a formulation of the *(two) wise men puzzle.*

Example 1. (The (two) wise men puzzle) The formulation is the following:

(1) $\Box(ws(a) :- bs(b))$.
(2) $\Box(ws(b) :- bs(a))$.
(3) $\Box([a]bs(b) :- bs(b))$.
(4) $\Box([b]bs(a) :- bs(a))$.
(5) $\Box([b]ws(b) :- [b]((bs(b)) \supset false))$.
(6) $\Box([a]ws(a) :- [a]((bs(a)) \supset false))$

In this example, $false$ is a distinguished symbol representing falsity; $bs(a)$ ($ws(a)$) represents the fact that a has a black (white) spot on his forehead. Clause (1) and (2) say that at least one wise man has a white spot; clause (3) and (4) say that if a wise man has a black spot then the other wise man knows that he has a black spot. Clause (5) and (6) can be read: if a wise man knows that he has not the black spot, then he knows that he has the white spot. Finally, modal operator $\Box$ in front of all clauses (1)–(6) defines them as common knowledge of agents a and b.

Assume that a is the first which is asked if he has a white spot on his forehead, that he does not know the answer and that b knows that (i.e., $[b](false :- [a]ws(a))$). When the wise man b is asked, he can answer yes (b knows that he has the white spot). Hence the following goal G

$$\Box(([b](false :- [a]ws(a))) \supset [b]ws(b))$$

is a consequence of the program. □

In Example 1, while the modal operators $[a]$, $[b]$ and $\Box$ give a way of distinguishing among information of the single agents and information common to all of them, embedded implications allow forms of hypothetical reasoning to be

[1] The modal logic on which our language is based contains all classical tautologies. Since we aim at defining a sound and complete proof procedure for our language, we cannot allow clauses like $a :- a \supset b$ in the language. In fact, while a is a consequence in classical logic of the formula $(a \supset b) \supset a$, a cannot be proved from $a :- a \supset b$ using a goal directed proof procedure, unless it makes use of a "restart rule" (see [10] for a study of goal directed proof procedures with restart for classical logic).

performed. It will be useful to note that, since all modalities are distributive with respect to conjunction, we can assume the *general form* $\Gamma_b(\Gamma_h A :- G)$ for clauses of $\mathcal{L}$, where Γ_b, Γ_h are arbitrary sequences of modalities (including the empty one).

Following [3], in our language the modal operators $[a_i]$ are all ruled by the axioms of K, while the operator $\Box$ is ruled by the axioms of $S4$. Thus, the axiomatization contains the following axiom schemas

$$\begin{array}{ll} (K_{[a_i]}) & [a_i](B \supset A) \supset ([a_i]B \supset [a_i]A) \\ (K_{\Box}) & \Box(B \supset A) \supset (\Box B \supset \Box A) \\ (T_{\Box}) & \Box A \supset A \\ (4_{\Box}) & \Box A \supset \Box\Box A \end{array}$$

Moreover, since the operator $\Box$ is intended to represent what is known by all agents, it interacts with each operator $[a_i]$ through the following *interaction axioms*

$$(I_{\Box,[a_i]}) \ \Box A \supset [a_i]A$$

As already remarked in [3] our modal operators $[a_i]$ and $\Box$ are quite similar to the notions of knowledge and common knowledge introduced in [14, 12]. However, our modality $\Box$ can be taken as a weaker version of common knowledge operator in [14, 12]. In particular the formula $A \wedge \Box(A \supset [a_i]A \wedge \ldots \wedge [a_k]A) \supset \Box A$ (the *induction axiom* for common knowledge) does not hold, while it is expected to hold when $\Box$ is the common knowledge operator. So, $\Box$ expresses only what any agent knows, and any agent knows that others know (and so on). In [3, 5] we have defined both *Kripke semantics* and a *sequent calculus* for modal language $\mathcal{L}$ containing the logical connectives $\neg$, $\wedge$, $\supset$, quantifiers $\exists$ and $\forall$, and the modalities $[a_1], \ldots, [a_k]$ and $\Box$.[2]

2.2 Operational Semantics of $\mathcal{L}$

In this section we will define operational derivability of a closed goal $(\exists\overline{X})G$ (with $\overline{X}$ the list of all free variables of G) from a program P by induction on the structure of G. Since the language allows modal operators, and in particular free occurrence of modalities in front of a goal, in the operational semantics we will introduce a notion of operational derivability of a closed goal $(\exists\overline{X})G$ from a program P in a certain *modal context*. A modal context is a sequence of modal operators $L_1|\ldots|L_n$.

To deal with embedded implications, each modal operator L_j in the sequence is associated with a (possibly empty) set of clauses D_j. In [13] the modal context has the simpler form of a list of sets of atoms $D_1|\ldots|D_n$ which are used for the only purpose of recording the ordering of the hypothetical updates of the D_i's during the computation.

To deal with both modalities and embedded implications we define the operational derivability of a closed goal G from a list of pairs $(L_0, D_0) \mid (L_1, D_1) \mid \ldots \mid (L_n, D_n)$ which keeps trace of embedded implications and modalized goals in a run of a proof: $L_1, \ldots, L_n$ are modalities, $D_1, \ldots, D_n$ are sets of clauses, D_0 is the initial program P, and L_0 (introduced for uniformity) is not used.

Following [16], in order to avoid problems with variable renaming and substitutions, given a set of clauses D, we denote by $[D]$ the set of all ground instances of the clauses in D. Moreover, in the following, we define the operational

[2] In particular, without loss of generality, we can take Herbrand domain as the constant domain for our Kripke interpretation (see [4] for more details).

derivability of a closed goal $(\exists\overline{X})G$ from a modal context as the operational derivability of goal $G[\overline{X}/\overline{t}]$, for some list of ground terms $\overline{t}$. The definition is by induction on the structure of the goal G.

Definition 1 (Operational Semantics of $\mathcal{L}$). A closed goal $(\exists\overline{X})G$ is *operationally derivable* from a modal context $(L_0, D_0) \mid (L_1, D_1) \mid \ldots \mid (L_n, D_n)$ if, for some list of ground terms $\overline{t}$, it holds that $\langle (L_0, D_0) \mid (L_1, D_1) \mid \ldots \mid (L_n, D_n)\rangle \vdash_{\mathcal{L}} G[\overline{X}/\overline{t}]$. The derivability of the ground goal $G[\overline{X}/\overline{t}]$ in turn is defined by induction on the structure of $G[\overline{X}/\overline{t}]$ as follows:

1. $\langle (L_0, D_0) \mid (L_1, D_1) \mid \ldots \mid (L_n, D_n)\rangle \vdash_{\mathcal{L}} T$;
2. $\langle (L_0, D_0) \mid (L_1, D_1) \mid \ldots \mid (L_n, D_n)\rangle \vdash_{\mathcal{L}} A$ if for some $j \in \{0, 1, \ldots, n\}$, there is a clause $\Gamma_b(\Gamma_h A :\!- G)$[3] $\in [D_j]$ and a k, $j \leq k \leq n$, such that
   Γ_b matches $\Gamma_1 = L_{j+1} \ldots L_k$, Γ_h matches $\Gamma_2 = L_{k+1} \ldots L_n$, and
   $\langle (L_0, D_0) \mid (L_1, D_1) \mid \ldots \mid (L_k, D_k)\rangle \vdash_{\mathcal{L}} G$;
3. $\langle (L_0, D_0) \mid (L_1, D_1) \mid \ldots \mid (L_n, D_n)\rangle \vdash_{\mathcal{L}} G_1 \ , \ G_2$ if
   $\langle (L_0, D_0) \mid (L_1, D_1) \mid \ldots \mid (L_n, D_n)\rangle \vdash_{\mathcal{L}} G_1$ and
   $\langle (L_0, D_0) \mid (L_1, D_1) \mid \ldots \mid (L_n, D_n)\rangle \vdash_{\mathcal{L}} G_2$;
4. $\langle (L_0, D_0) \mid (L_1, D_1) \mid \ldots \mid (L_n, D_n)\rangle \vdash_{\mathcal{L}} L_{n+1}G$ if
   $\langle (L_0, D_0) \mid (L_1, D_1) \mid \ldots \mid (L_n, D_n) \mid (L_{n+1}, \emptyset)\rangle \vdash_{\mathcal{L}} G$
5. $\langle (L_0, D_0) \mid (L_1, D_1) \mid \ldots \mid (L_n, D_n)\rangle \vdash_{\mathcal{L}} L_{n+1}(D \supset G)$ if
   $\langle (L_0, D_0) \mid (L_1, D_1) \mid \ldots \mid (L_n, D_n) \mid (L_{n+1}, D)\rangle \vdash_{\mathcal{L}} G$.

In the definition above, the rules for the distinguished proposition T, for conjunctions and for goals are the usual ones. The rules 4) and 5) for embedded implications and modalized goals are quite similar. Embedded implications are always preceded by at least one modal operator. To prove a goal $L(D \supset G)$, where L is an arbitrary modal operator, the modal operator L is added to the current context together with the clauses in D, and G is proved in the resulting context. Similarly a goal LG is proved by adding the modal operator L to the current context (with associated empty set of clauses) and proving G in the resulting context.

To prove an atomic goal A a clause is selected in the initial program D_0 or from a set of clauses D_j that has been added in the context, by proving some implication goal. In both cases, to verify that a clause is applicable in the current context, it must be checked whether the modal operators in the clause (both in front of it and in front of its head) match the current context, from L_{j+1} to L_n. In particular, the sequence of the modal operators Γ_b in front of the selected clause must match a prefix of the sequence $L_{j+1} \ldots L_n$, while the sequence of the modal operators Γ_h in front of the head of the selected clause must match the remaining part of the sequence. We say that a sequence Γ of modal operators matches another sequence Γ' if each modal operator in Γ matches a sequence of operators in Γ' in the ordering. More precisely, each modal operator $[a_i]$ in Γ may only match the operator $[a_i]$ itself, while each operator $\Box$ in Γ may match either an empty sequence or an arbitrary sequence of modalities in Γ'.

Definition 2 (Matching Operation for $\mathcal{L}$). Let $\Gamma = L_1 \ldots L_r$ and Γ' be two sequences of modalities, then we say that Γ *matches* Γ' if there exist r functions $f_1, \ldots, f_r$ such that the following conditions hold:

- $f_j([a_i]) = [a_i]$, for all $j = 1, \ldots, r$ and for all modal operators $[a_i]$;

[3] For simplicity, we will represent facts H as clauses $H :\!- T$.

- $f_j(\Box)$ is any sequence of modalities (including the empty one), for all $j = 1, \ldots, r$;
- $f_1(L_1) \ldots f_r(L_r) = \Gamma'$.

Given a program P and a closed goal G, we say that G is provable from P, if $\langle (L_0, P) \rangle \vdash_{\mathcal{L}} G$ can be derived by applying the operational rules above.

Example 2. (The (two) wise men puzzle) Given the formulation of the *(two) wise men puzzle* of Example 1, the goal $\Box(([b](false :\!\!- [a]ws(a))) \supset [b]ws(b))$ has the following derivation:

$$\begin{array}{l}
\langle (L_0, P) \rangle \vdash_{\mathcal{L}} \Box(([b](false :\!\!- [a]ws(a))) \supset [b]ws(b)) \\
\langle (L_0, P) \mid (\Box, [b](false :\!\!- [a]ws(a)))\rangle \vdash_{\mathcal{L}} [b]ws(b) \\
\langle (L_0, P) \mid (\Box, [b](false :\!\!- [a]ws(a))) \mid ([b], \emptyset)\rangle \vdash_{\mathcal{L}} ws(b) \\
\langle (L_0, P) \mid (\Box, [b](false :\!\!- [a]ws(a)))\rangle \vdash_{\mathcal{L}} [b]((bs(b)) \supset false) \text{ by (5)} \\
\langle (L_0, P) \mid (\Box, [b](false :\!\!- [a]ws(a))) \mid ([b], bs(b))\rangle \vdash_{\mathcal{L}} false \\
\langle (L_0, P) \mid (\Box, [b](false :\!\!- [a]ws(a))) \mid ([b], bs(b))\rangle \vdash_{\mathcal{L}} [a]ws(a) \text{ by (7)} \\
\langle (L_0, P) \mid (\Box, [b](false :\!\!- [a]ws(a))) \mid ([b], bs(b)) \mid ([a], \emptyset)\rangle \vdash_{\mathcal{L}} ws(a) \\
\langle (L_0, P) \mid (\Box, [b](false :\!\!- [a]ws(a))) \mid ([b], bs(b)) \mid ([a], \emptyset)\rangle \vdash_{\mathcal{L}} bs(b) \text{ by (1)} \\
\langle (L_0, P) \mid (\Box, [b](false :\!\!- [a]ws(a))) \mid ([b], bs(b))\rangle \vdash_{\mathcal{L}} bs(b) \text{ by (3)} \\
\langle (L_0, P) \mid (\Box, [b](false :\!\!- [a]ws(a))) \mid ([b], bs(b))\rangle \vdash_{\mathcal{L}} T \text{ by } (5'').
\end{array}$$

where (7) and $(5'')$ are the two clauses added to the modal context during the computation, i.e. respectively $[b](false :\!\!- [a]ws(a))$ and $bs(b)$. □

3 The first step of the translation

The aim of this section is to introduce the first step of the translation and its target language. The goal of this step is to eliminate embedded implications from programs and queries in the language $\mathcal{L}$, and to transform extended clauses and goals in Horn clauses and goals which allow only modalities to occur freely in front of clauses, in front of clause heads and in front of goals.

The idea is to introduce some new agents, one for each goal $L(D \supset G)$, and to associate the set of clauses D to it as its beliefs. After this, we can replace each goal $L(D \supset G)$ by a goal which asks if the introduced agent believes G. Each new agent will be represented by a new modality of type K. We will denote the new modality introduced for $L(D \supset G)$ by L^c, where c is a new constant, so that clauses D can only be used at the point where they have been extracted. Before showing the procedure to transform programs and goals, let us see an example.

Example 3. (Extracting an embedded implication from a clause) Let

$$C \equiv \Gamma_b(\Gamma_h A :\!\!- G_1, \ldots, [a_i](D_j \supset G_j), \ldots, G_m)$$

be a clause of a program P in language $\mathcal{L}$, then our goal is to transform C in a new clause C' without implication goal $[a_i](D_j \supset G_j)$. Let us introduce a new modal operator of type K, $[a_i]^c$, where c is a constant which is never used in the program P. Then C' will be the clause

$$\Gamma_b(\Gamma_h A :\!\!- G_1, \ldots, [a_i]^c G_j, \ldots, G_m)$$

and we could modify P by replacing C with C' and by adding the definition

$$\Box[a_i]^c D_j$$

to the resulting program. Intuitively, the updated definition $\Box[a_i]^c D_j$ specifies the beliefs of the new agent c, and the presence of common knowledge operator $\Box$ in the definition above allows it to be visible through nested modal operators during a computation. The goal $[a_i]^c G_j$ allows to ask if G_j belongs to beliefs of c. The goal $[a_i]^c G_j$ plays the role of embedded implication $[a_i](D_j \supset G_j)$, where D_j is the set of clauses associated with agent c. The difference is that, within the embedded implication, D_j is *local* to G_j and only goal G_j can use it; vice versa in $\Box[a_i]^c D_j$, D_j is not local to any goals (the difference is similar to that between *blocks* and *modules* in imperative languages).

Nevertheless, if the modal operator $[a_i]^c$ is only used in the goal $[a_i]^c G_j$ and, for defining beliefs of c, in $\Box[a_i]^c D_j$, then the goals $[a_i]^c G_j$ and $[a_i](D_j \supset G_j)$ have precisely the same meaning. The first step of translation uses this idea to eliminate embedded implications from programs in the language $\mathcal{L}$. $\Box$

It is worth noting that embedded implications must be also eliminated from queries. We will deal with programs and queries uniformly by introducing a fictitious goal. That is, let P be a program and G a goal, to ask if a ground instance of G is derivable from P is equivalent to ask if a ground istance of $\bot(X_1, \ldots, X_s)$ is derivable from P updated with the clause $\bot(X_1, \ldots, X_s) \mathrel{:-} G$, where $\bot$ is a new predicative symbol and $X_1, \ldots, X_s$ are all free variables of G.

Now we are ready to introduce our procedure for extracting embedded implications from a pair *program* and *query*. The functions $|.|_p$ and $|.|_g$ are for distributing modalities through goals and clauses and are defined as follows:

$$|\Gamma(D_1 \ . \ D_2)|_p = |\Gamma D_1|_p \cup |\Gamma D_2|_p \qquad |\Gamma(G_1 \ , \ G_2)|_g = |\Gamma G_1|_g \ , \ |\Gamma G_2|_g$$
$$|\Gamma_b(\Gamma_h A \mathrel{:-} G)|_p = \Gamma_b(\Gamma_h A \mathrel{:-} |G|_g) \qquad |\Gamma A|_g = \Gamma A$$
$$|\Gamma(D \supset G)|_g = \Gamma(|D|_p \supset |G|_g)$$

with Γ, Γ_b and Γ_h any sequence of modalities.

Definition 3 (Procedure for Extracting Embedded Implications). Let P be a program and G a goal. Then the procedure in Fig. 1 takes as input the pair P and G and returns as output P^e, the program obtained by extracting embedded implications from P and G, where the function *gen_symbol* returns a

---

begin
  $S := |P \cup \{\bot(\overline{X}) \mathrel{:-} G\}|_p$;
  while ($\exists$ a clause $C \equiv \Gamma_b(\Gamma_h A \mathrel{:-} G_1, \ldots, \Gamma_g L(D_j \supset G_j), \ldots, G_m)$ in S) **do**
    begin
      $c := gen\_symbol$;
      $C' := \Gamma_b(\Gamma_h A \mathrel{:-} G_1, \ldots, |\Gamma_g L^c G_j|_g, \ldots, G_m)$;
      $S := (S - \{C\}) \cup \{C'\} \cup |2L^c D_j|_p$
    end;
  $P^e := S$;
end

Fig. 1. Procedure for Extracting Embedded Implications.

new constant never used before, so that L^c is a new modality of type K for each step, $\bot$ is a new predicative symbol and $\overline{X}$ the list of all free variables of G.

We denote with $\mathcal{L}^e$ the language $\mathcal{L}$ extended with a (finite) set of new modal operators L^c of type K, where L is a modality $\Box$ or $[a_i]$ and c is a constant. $\mathcal{L}^e$ is the language in which the procedure in Fig. 1 translates programs and goals. Note that translated programs and goals do not contain embedded implications.

The matching operation for $\mathcal{L}^e$ must be extended with respect to that of $\mathcal{L}$ to take into account the new modal operators. In Example 3, we have introduced a new modality $[a_i]^c$, so that only goal G_j can exploit the clauses in D_j. We now want to compare the computation of goal $[a_i](D_j \supset G_j)$ and that of the goal $[a_i]^c G_j$ obtained from the translation step above. Let us suppose that $[a_i](D_j \supset G_j)$ is operationally derivable from a modal context $(L_0, D_0)|(L_1, D_1)|\ldots|(L_n, D_n)$, and G_j is an atomic formula A. Following Definition 1, we have

$$\langle (L_0, D_0) \mid (L_1, D_1) \mid \ldots \mid (L_n, D_n) \rangle \vdash_{\mathcal{L}} [a_i](D_j \supset A)$$
$$\langle (L_0, D_0) \mid (L_1, D_1) \mid \ldots \mid (L_n, D_n) \mid ([a_i], D_j) \rangle \vdash_{\mathcal{L}} A$$
for some $j = 0, \ldots, n$, there is a clause $\Gamma_b(\Gamma_h A :- G) \in [D_j]$
and a k, $j \leq k \leq n+1$, such that
Γ_b matches $\Gamma_1 = L_{j+1} \ldots L_k$, Γ_h matches $\Gamma_2 = L_{k+1} \ldots L_n[a_i]$,
and $\langle (L_0, D_0) \mid (L_1, D_1) \mid \ldots \mid (L_k, D_k) \rangle \vdash_{\mathcal{L}} G$

Thus, by Definition 2, since Γ_h *matches* $L_{k+1} \ldots L_n[a_i]$, then Γ_h has the form $\Gamma'_h[a_i]$. Let us now consider the goal $[a_i]^c G_j$

$$\langle (L_0, D_0) \mid (L_1, D_1) \mid \ldots \mid (L_n, D_n) \rangle \vdash_{\mathcal{L}} [a_i]^c A$$
$$\langle (L_0, D_0) \mid (L_1, D_1) \mid \ldots \mid (L_n, D_n) \mid ([a_i]^c, \emptyset) \rangle \vdash_{\mathcal{L}} A.$$

The above selected clause $\Gamma_b(\Gamma_h A :- G)$ is not applicable to prove A, because $\Gamma_h \equiv \Gamma'_h[a_i]$ does not match $L_{k+1} \ldots L_n[a_i]^c$, since $[a_i]$ does not match $[a_i]^c$. In other words, to move a goal $[a_i]^c G_j$ we cannot use the believes of agent a_i, as we would expect. To avoid this problem we must allow each modality L to match L^c, for all constant c, and modal operator $\Box$ to match all modalities. Hence, we introduce the following definition of matching for the language $\mathcal{L}^e$.

Definition 4 (Matching Operation for $\mathcal{L}^e$). Let $\Gamma = L_1 \ldots L_r$ and Γ' be two sequence of modalities, then we say that Γ *matches* Γ' if there exist r functions $f_1, \ldots, f_r$ such that the following conditions hold:

- $f_j([a_i]) = [a_i]$ or $[a_i]^c$, $j = 1, \ldots, r$, for all modal operators $[a_i]$ and for all constant c;
- $f_j(L^c) = L^c$, $j = 1, \ldots, r$, for all modal operators L and for all constant c;
- $f_j(\Box)$, $j = 1, \ldots, r$, is any sequence of modalities (including the empty one);
- $f_1(L_1) \ldots f_r(L_r) = \Gamma'$.

Definition 5 (Operational Semantics of $\mathcal{L}^e$). Operational semantics $\vdash_{\mathcal{L}^e}$ for the language $\mathcal{L}^e$ is defined as $\vdash_{\mathcal{L}}$ of Definition 1, apart from the *matching operation* which is given in Definition 4.

Note that $\vdash_{\mathcal{L}^e}$ subsumes $\vdash_{\mathcal{L}}$. However, if the program and the goal do not contain embedded implications (as it is the case), then $D_j = \emptyset$, for all $j = 1, \ldots, n$, in all modal contexts of derivations, and rule 5) of Definition 1 is never used in $\vdash_{\mathcal{L}^e}$.

The axiomatization of Sect. 2.1 for language $\mathcal{L}$ must be modified for $\mathcal{L}^e$. In particular, the modal operators $[a_i]$ and the new modal operators L^c are all ruled by axioms of K. Moreover, since we wish that clauses in updated definition $\Box[a_i]^c D$ can use believes of agent a_i, the axiom $[a_i]A \supset [a_i]^c A$ must be included for all $[a_i]$ and for all constants c. Finally, the set of interaction axioms for modality $\Box$ also contains the axioms $\Box A \supset L^c A$ for all modalities L^c.

Example 4. (First step of translation on the (two) wise men puzzle) Given the program P and the goal G of the Example 1, then after applying the procedure for extracting embedded implications (Definition 3), we will obtain the following program P^e:

(1) $\Box(ws(a) :- bs(b))$.
(2) $\Box(ws(b) :- bs(a))$.
(3) $\Box([a]bs(b) :- bs(b))$.
(4) $\Box([b]bs(a) :- bs(a))$.
(5′) $\Box([b]ws(b) :- [b]^{c_1} false)$.
(5″) $\Box[b]^{c_1}(bs(b))$.
(6′) $\Box([a]ws(a) :- [a]^{c_2} false)$.
(6″) $\Box[a]^{c_2}(bs(a))$.
(7) $\Box\Box^{c_3}[b](false :- [a]ws(a))$.
(8) $\bot :- \Box^{c_3}[b]ws(b)$.

Note that clauses (7), (5″) and (6″) have been added. The first one by extracting the embedded implication of goal G, and the other ones by extracting the embedded implications from clauses (5) and (6) of the program P. Clauses (5′) and (6′) in P^e replace (5) and (6) in P. Finally, clause (8) represents the new goal. The derivation of $\bot$ from P^e mimics the derivation in Example 2:

$\langle(L_0, P^e)\rangle \vdash_{\mathcal{L}^e} \bot$
$\langle(L_0, P^e)\rangle \vdash_{\mathcal{L}^e} \Box^{c_3}[b]ws(b)$ by (8)
$\langle(L_0, P^e) \mid (\Box^{c_3}, \emptyset)\rangle \vdash_{\mathcal{L}^e} [b]ws(b)$
$\langle(L_0, P^e) \mid (\Box^{c_3}, \emptyset) \mid ([b], \emptyset)\rangle \vdash_{\mathcal{L}^e} ws(b)$
$\langle(L_0, P^e) \mid (\Box^{c_3}, \emptyset)\rangle \vdash_{\mathcal{L}^e} [b]^{c_1} false$ by (5′)
$\langle(L_0, P^e) \mid (\Box^{c_3}, \emptyset) \mid ([b], \emptyset) \mid ([b]^{c_1}, \emptyset)\rangle \vdash_{\mathcal{L}^e} false$
$\langle(L_0, P^e) \mid (\Box^{c_3}, \emptyset) \mid ([b], \emptyset) \mid ([b]^{c_1}, \emptyset)\rangle \vdash_{\mathcal{L}^e} [a]ws(a)$ by (7)
$\langle(L_0, P^e) \mid (\Box^{c_3}, \emptyset) \mid ([b], \emptyset) \mid ([b]^{c_1}, \emptyset) \mid ([a], \emptyset)\rangle \vdash_{\mathcal{L}^e} ws(a)$
$\langle(L_0, P^e) \mid (\Box^{c_3}, \emptyset) \mid ([b], \emptyset) \mid ([b]^{c_1}, \emptyset) \mid ([a], \emptyset)\rangle \vdash_{\mathcal{L}^e} bs(b)$ by (1)
$\langle(L_0, P^e) \mid (\Box^{c_3}, \emptyset) \mid ([b], \emptyset) \mid ([b]^{c_1}, \emptyset)\rangle \vdash_{\mathcal{L}^e} bs(b)$ by (3)
$\langle(L_0, P^e) \mid (\Box^{c_3}, \emptyset) \mid ([b], \emptyset) \mid ([b]^{c_1}, \emptyset)\rangle \vdash_{\mathcal{L}^e} T$ by (5″)

□

The correctness of the first step of translation is based on Lemma 6. It corresponds to a step of transformation of procedure in Fig. 1 in the operational semantics.

Lemma 6 (Extracting an implication from a clause). *Let* $(L_0, D_0)|(L_1, D_1)| \ldots |(L_n, D_n)$ *be a modal context and* G *a goal, then*

$$\langle(L_0, D_0)| \ldots |(L_n, D_n)\rangle \vdash_{\mathcal{L}^e} G \text{ iff } \langle(L_0, D_0' \cup \Box L^c D_j)| \ldots |(L_n, D_n)\rangle \vdash_{\mathcal{L}^e} G$$

where:

- $C \equiv \Gamma_b(\Gamma_h A :- G_1, \ldots, \Gamma_g L(D_j \supset G_j), \ldots, G_m)$ *is a clause of* D_0;
- $C' \equiv \Gamma_b(\Gamma_h A :- G_1, \ldots, \Gamma_g L^c G_j, \ldots, G_m)$;
- $D_0' = (D_0 - \{C\}) \cup \{C'\}$;
- L^c *is a new modality used only in* C' *and for defining* $\Box L^c D_j$.

The proof of this lemma, omitted for lack of space, is done by induction on lenght of the derivation of G. The equivalent relation between pairs (P, G) and $(P^e, \bot)$ of Definition 3 is given by the following theorem.

Theorem 7 (Soundness and Completeness of $\vdash_{\mathcal{L}^e}$ w.r.t. $\vdash_{\mathcal{L}}$). *Let P be a program and $(\exists\overline{X})G$ a goal in the language $\mathcal{L}$, with $\overline{X}$ the list of all free variables of G, then $P \vdash_{\mathcal{L}} G[\overline{X}/\overline{t}]$ iff $P^e \vdash_{\mathcal{L}^e} \bot(\overline{X})[\overline{X}/\overline{t}]$, for some list of ground terms $\overline{t}$, where P^e is the new program obtained from P and G as in Definition 3.*

Proof. We prove that $\{P \vdash_{\mathcal{L}} G[\overline{X}/\overline{t}] \iff S \vdash_{\mathcal{L}^e} \bot(\overline{X})[\overline{X}/\overline{t}]\}$ is a while-loop invariant for the procedure in Fig. 1. At the end of procedure, P^e will be setted with the final values of S. Moreover, the procedure terminates, since the number of implications $\supset$ in P is finite and it decreases at each step. Let us suppose that $\{P \vdash_{\mathcal{L}} G[\overline{X}/\overline{t}] \iff S \vdash_{\mathcal{L}^e} \bot(\overline{X})[\overline{X}/\overline{t}]\}$ holds at the beginning of the loop. We prove that $\{P \vdash_{\mathcal{L}} G[\overline{X}/\overline{t}] \iff S' \vdash_{\mathcal{L}^e} \bot(\overline{X})[\overline{X}/\overline{t}]\}$ holds at the end of the loop, where $S' = (S - \{C\}) \cup \{C'\} \cup |\Box L^c D_j|_p$, $C = \Gamma_b(\Gamma_h A :- G_1, \ldots, \Gamma_g L(D_j \supset G'_j), \ldots, G_m)$ and $C' = \Gamma_b(\Gamma_h A :- G_1, \ldots, |\Gamma_g L^c G'_j|_g, \ldots, G_m)$. Thus, we have to show that $\langle (L_0, S)\rangle \vdash_{\mathcal{L}^e} \bot(\overline{X})[\overline{X}/\overline{t}]$ if and only if $\langle (L_0, S')\rangle \vdash_{\mathcal{L}^e} \bot(\overline{X})[\overline{X}/\overline{t}]$, and this is true by Lemma 6. □

4 The second step of the translation

In this section we will show how the language $\mathcal{L}^e$ can be translated into a set of Horn clauses, which can then be executed as a standard logic program.

The translation methods is based on the idea of implementing directly the operational semantics making explicit reference to the modal context. This is achieved by adding to all predicates an extra argument representing the modal context where the predicate must hold. In particular, a modal context allows us to record the ordering between modalities found in front of goals, during a computation. Note that the notion of modal context plays a role similar to that of *prefixes* of formulas in Fitting's tableaux system (see [9]). Intuitively, a prefix is a name for a possible world, and the same is for a modal context. A prefix allows us to recognize syntactically whether the worlds being named are accessible or not.

First of all, it is worth noting that after applying first step of translation, since a program and a goal do not contain embedded implication, we can simplify the notation of operational semantics of Definition 5:

Definition 8. (Operational Semantics of $\mathcal{L}^e$ without embedded implication) Given a program P and a modal context $L_1 \mid L_2 \mid \ldots \mid L_n$, operational derivability of a ground goal $G[\overline{X}/\overline{t}]$ from P in the modal context $L_1 \mid L_2 \mid \ldots \mid L_n$, that is $\langle P, L_1 \mid L_2 \mid \ldots \mid L_n\rangle \vdash_{\mathcal{L}^e} G[\overline{X}/\overline{t}]$, is defined by induction on the structure of $G[\overline{X}/\overline{t}]$ as follows:

1. $\langle P, L_1 \mid L_2 \mid \ldots \mid L_n\rangle \vdash_{\mathcal{L}^e} T$;
2. $\langle P, L_1 \mid L_2 \mid \ldots \mid L_n\rangle \vdash_{\mathcal{L}^e} A$ if there is a clause $\Gamma_b(\Gamma_h A :- G) \in [P]$ and a k, $0 \leq k \leq n$, such that
   Γ_b matches $L_1 \ldots L_k$, Γ_h matches $L_{k+1} \ldots L_n$, and
   $\langle P, L_1 \mid L_2 \mid \ldots \mid L_k\rangle \vdash_{\mathcal{L}^e} G$;
3. $\langle P, L_1 \mid L_2 \mid \ldots \mid L_n\rangle \vdash_{\mathcal{L}^e} G_1\ ,\ G_2$ if
   $\langle P, L_1 \mid L_2 \mid \ldots \mid L_n\rangle \vdash_{\mathcal{L}^e} G_1$ and $\langle P, L_1 \mid L_2 \mid \ldots \mid L_n\rangle \vdash_{\mathcal{L}^e} G_2$;
4. $\langle P, L_1 \mid L_2 \mid \ldots \mid L_n\rangle \vdash_{\mathcal{L}^e} L_{n+1} G$ if

$\langle P, L_1 \mid L_2 \mid \ldots \mid L_n \mid L_{n+1} \rangle \vdash_{\mathcal{L}^e} G$.

To prove a goal G from a program P means to show that G is operationally derivable from P in the empty modal context ε, that is $\langle P, \varepsilon \rangle \vdash_{\mathcal{L}^e} G$.

Moreover, since functions $|.|_p$ and $|.|_g$ distribute modalities through conjunctions of goals and clauses, clauses after the first step of the translation will have the general form

$$\Gamma_b(\Gamma_h A_0 :- \Gamma_{g_1} A_1, \ldots, \Gamma_{g_m} A_m) \tag{2}$$

where $A_{i=0,1,\ldots,m}$ are atomic predicates and $\Gamma_b, \Gamma_h, \Gamma_{g_1}, \ldots, \Gamma_{g_m}$ arbitrary sequences of modalities. Thus, by combining rules 2), 3) and 4) of Definition 8

rule 2′) $\langle P, L_1 \mid L_2 \mid \ldots \mid L_n \rangle \vdash_{\mathcal{L}^e} A$
if there is a clause $C \equiv \Gamma_b(\Gamma_h A :- \Gamma_{g_1} A_1, \ldots, \Gamma_{g_m} A_m) \in [P]$
and a k, $0 \leq k \leq n$, such that
Γ_b matches $L_1 \ldots L_k$, Γ_h matches $L_{k+1} \ldots L_n$, and
$\langle P, L_1 \mid L_2 \mid \ldots \mid L_k \mid \Gamma_{g_1} \rangle \vdash_{\mathcal{L}^e} A_1$,
$\ldots$
$\langle P, L_1 \mid L_2 \mid \ldots \mid L_k \mid \Gamma_{g_m} \rangle \vdash_{\mathcal{L}^e} A_m$.

Intuitively, if a modal context is a name of a world, then rule 2′) is equivalent to ask if A is true in the world $L_1 \ldots L_n$ using C, which is true in the initial world. Now, the world $L_1 \ldots L_n$ is accessible from $L_1 \ldots L_k$ through the path $L_{k+1}| \ldots |L_n$. Thus, if the sequence of modalities Γ_b and Γ_h allow *to go through* the path $L_1| \ldots |L_k$ and $L_{k+1}| \ldots |L_n$, and the body of C is true in the world $L_1 \ldots L_k$, then A holds in $L_1 \ldots L_n$. The body of C is true in the world $L_1 \ldots L_k$ if each subgoals A_i is true in the world $L_1 \ldots L_k \Gamma_{g_i}$. A sequence of modalities Γ allows the formula α *to go through* a path $L_{j_1}| \ldots |L_{j_k}$ if Γ *matches* the path $L_{j_1}| \ldots |L_{j_k}$, according to the properties of modal operators.

The idea of second step of translation for eliminating modalities is based on adding to all atomic predicates an argument which represents the name of world where the predicates have to be proved. In others worlds, to move the modal context of operational semantics directly into the predicates.

Let $match(\Gamma_b, \Gamma_h, X, Y)$ be a predicate such that it has success if the joint sequence of modalities Γ_b and Γ_h matches the current context X, according to Definition 4, and it returns the matched sequence of X by Γ_b in Y. So (2) can be translated as

$$A_0(X) :- match(\Gamma_b, \Gamma_h, X, Y), A_1(Y \circ \Gamma_{g_1}), \ldots, A_m(Y \circ \Gamma_{g_m})^4 \tag{3}$$

obtaining a Horn clause, and operational derivability will be defined as SLD resolution. In particular, let $\Gamma_g A$ be a subgoal in the body of a clause, we can translate it in $A(Y \circ \Gamma_g)$, where Y is a variable which is unified with the current context (the name of the world where $\Gamma_g A$ has to be proved) and linked (denoted by "$\circ$") with Γ_g for proving A.

Finally, note that the added argument "X" will always be ground during the computation. In fact, since we ask to prove a query in the empty initial modal context, we start each resolution with a goal as $A(\Gamma)$, where Γ does not contain variables. Thus, it is not possible to introduce variables into resolvent, so the *match operation* works correctly as above described.

We can now give the procedure for translating goals and extended clauses of the language $\mathcal{L}^e$ into first order logic, by eliminating modal operators.

[4] X and Y do not belong to the set of variables of clause (2).

Definition 9 (Procedure for Translating into Horn Clauses Language). Let P be a program and G a goal of language $\mathcal{L}$, let P^e be the program obtained applying the procedure of Definition 3 to P and G, then the procedure in Fig. 2 takes as input P^e and returns as output P^{tr}, the program obtained by translation of P^e into Horn clauses language, where the sequence $\Gamma' \circ \Gamma''$ is the concatena-

---

begin
  $S := P^e$;
  for (each clause $C \equiv \Gamma_b(\Gamma_h A :- \Gamma_{g_1} A_1, \ldots, \Gamma_{g_m} A_m)$ in S) **do**
    begin
      $C' := A(X) :- match(\Gamma_b, \Gamma_h, X, Y), A_1(Y \circ \Gamma_{g_1}), \ldots, A_m(Y \circ \Gamma_{g_m})$;
      $S := (S - \{C\}) \cup \{C'\}$
    end;
  $P^{tr} := S$;
end

Fig. 2. Procedure for Translating into Horn Clause Language.

tion of sequence Γ' the sequence Γ''. Moreover, if A is $p(t_1, \ldots, t_s)$, then $A(X)$ and $A(Y \circ \Gamma_{g_j})$ are $p(X, t_1, \ldots, t_s)$ and $p(Y \circ \Gamma_{g_j}, t_1, \ldots, t_s)$, respectively. Finally, the predicate $match/4$ carries out the matching operation of Definition 5, and X and Y are variables.

Let us see how the second step of translation works on the program P^e of the Example 4.

Example 5. (Second step of translation on the (two) wise men puzzle) Given the program P^e of the Example 4, after applying the procedure of Definition 9, we will obtain the following program P^{tr} (we will denote with ε the empty sequence of modalities):

(1) $ws(X, a) :- match(\Box, \varepsilon, X, Y), bs(Y \circ \varepsilon, b)$.
(2) $ws(X, b) :- match(\Box, \varepsilon, X, Y), bs(Y \circ \varepsilon, a)$.
(3) $bs(X, b) :- match(\Box, [a], X, Y), bs(Y \circ \varepsilon, b)$.
(4) $bs(X, b) :- match(\Box, [b], X, Y), bs(Y \circ \varepsilon, a)$.
(5′) $ws(X, b) :- match(\Box, [b], X, Y), false(Y \circ [b]^{c_1})$.
(5″) $bs(X, b) :- match(\Box[b]^{c_1}, \varepsilon, X, Y)$.
(6′) $ws(X, a) :- match(\Box, [a], X, Y), false(Y \circ [a]^{c_2})$.
(6″) $bs(X, a) :- match(\Box[a]^{c_2}, \varepsilon, X, Y)$.
(7) $false(X) :- match(\Box\Box^{c_3}[b], \varepsilon, X, Y), ws(Y \circ [a], a)$
(8) $\bot(X) :- match(\varepsilon, \varepsilon, X, Y), ws(\Box^{c_3}[b], b)$

and the goal $\bot(\varepsilon)$ succeeds from P^{tr} with the following SLD derivation:

$P^{tr} \vdash \bot(\varepsilon)$
$P^{tr} \vdash match(\varepsilon, \varepsilon, \varepsilon, Y_0), ws(Y_0 \circ \Box^{c_3}[b], b)$ by (8), $X_0 = \varepsilon$
$P^{tr} \vdash ws(\Box^{c_3}[b], b)$ with $Y_0 = \varepsilon$
$P^{tr} \vdash match(\Box, [b], \Box^{c_3}[b], Y_1), false(Y_1 \circ [b]^{c_1})$ by (5′), $X_1 = \Box^{c_3}[b]$
$P^{tr} \vdash false(\Box^{c_3}[b]^{c_1})$ with $Y_1 = \Box^{c_3}$

$P^{tr} \vdash match(\Box\Box^{c_3}[b], \varepsilon, \Box^{c_3}[b]^{c_1}, Y_2), ws(Y_2 \circ [a], a)$ by (7), $X_2 = \Box^{c_3}[b]^{c_1}$
$P^{tr} \vdash ws(\Box^{c_3}[b]^{c_1}[a], a)$ with $Y_2 = \Box^{c_3}[b]^{c_1}$
$P^{tr} \vdash match(\Box, \varepsilon, \Box^{c_3}[b]^{c_1}[a], Y_3), bs(Y_3 \circ \varepsilon, b)$ by (1), $X_3 = \Box^{c_3}[b]^{c_1}[a]$
$P^{tr} \vdash bs(\Box^{c_3}[b]^{c_1}[a], b)$ with $Y_3 = \Box^{c_3}[b]^{c_1}[a]$
$P^{tr} \vdash match(\Box, [a], \Box^{c_3}[b]^{c_1}[a], Y_4), bs(Y_4 \circ \varepsilon, a)$ by (3), $X_4 = \Box^{c_3}[b]^{c_1}[a]$
$P^{tr} \vdash bs(\Box^{c_3}[b]^{c_1}, a)$ with $Y_4 = \Box^{c_3}[b]^{c_1}$
$P^{tr} \vdash match(\Box[b]^{c_1}, \varepsilon, \Box^{c_3}[b]^{c_1}, Y_5)$ by $(5'')$, $X_5 = \Box^{c_3}[b]^{c_1}$
$P^{tr} \vdash T$ with $Y_5 = \Box^{c_3}[b]^{c_1}$

Again, notice that the steps of this derivation correspond to the step of the previous derivations. □

The correctness of the whole process of translation is given by the following theorem.

Theorem 10 (Correctness of the Translation). *Let P be a program and $(\exists\overline{X})G$ a goal in the language $\mathcal{L}$, with $\overline{X}$ the list of all free variables of G, then $P \vdash_{\mathcal{L}} G[\overline{X}/\overline{t}]$ iff $P^{tr} \vdash \bot(\varepsilon, \overline{X})[\overline{X}/\overline{t}]$, for some list of ground terms $\overline{t}$, where P^{tr} is the new program after applying procedures of Definition 3 and Definition 9 to P and G, and $\vdash$ is standard operational derivability relation for Horn clause language.*

5 Conclusions and related works

In this paper we have presented a technique for translating extended logic programs with multiple modalities and embedded implication into standard Prolog. This technique has been implemented and tested on several examples. Since the performance of the translated program heavily dependes on the predicate *match*, special care was devoted to its implementation.

As we have already pointed out, the approach of this paper could be easily modified to deal with modal operators with different properties. Since the properties of modal operators are described by the matching operation, it would be easy to modify predicate *match* accordingly.

Translation methods for modal logics have been developed by many authors [8] as an alternative approach to the development of specific theorem proving techniques and tools. In fact, by translating a modal theorem into predicate logic, it is possible to use a standard theorem prover without the need to build a new one.

The translation methods for modal logics are based on the idea of making explicit reference to the worlds by adding to all predicates an argument representing the world where the predicate holds, so that modal operators can be transformed in quantifiers of classical logic. In particular, in the functional approach [18, 2], accessibility is represented by means of functions: a modal operator $[m]$ is translated into $\forall F_m$, where F_m is a function of *sort* m, and the worlds will always be represented by a composition of functions, such as $F_{m_1} \circ \ldots \circ F_{m_n}$. For instance, the following clause of language $\mathcal{L}$

$$\Box([a]p :\!- [b]q)$$

will be translated into

$$\forall F_\Box((\forall G_a q(F_\Box \circ G_a)) :\!- (\forall H_b p(F_\Box \circ H_b)))$$

This translation is correct if the accessibility relation is assumed to be serial and if the domain of interpretations is constant. We can assume that these conditions hold in our case.

The above formula can be transformed to clausal form as follows

$$p(F_\Box \circ G_a) :- q(F_\Box \circ c_b)$$

where c_b is a Skolem constant of sort b. Note that, since the body is negated, all universally quantified variables in the body have to be skolemized.

The properties of the accessibility relation, such as reflexivity or transitivity, can usually be described with equations which can be translated into a theory unification algorithm. In our case, a variable $F_\Box$ can match any sequence of functions, whereas a variable F_m can match only a function of sort m.

An advantage of the functional method with respect to other translation methods is that it keeps the structure of the original formula. A variant of this approach is presented in [17].

It is easy to see that this approach closely corresponds to the second step of our translation. Sequences of functions in the functional approach correspond to sequences of modal operators in our case, Skolem constants correspond to the new modal operators, and the equational unification is performed by predicate *match* (in our case we do not need full unification, but only matching).

However it would not be possible to apply directly the translation method to a program in the language $\mathcal{L}$. In fact, given a clause of $\mathcal{L}$ containing embedded implication, the translation will produce a formula of predicate logic with the same structure. It is easy to see that, by transforming such a formula to clause form, does not yield, in general, a Horn clause.

Language $\mathcal{L}$ does not use quantifiers, and all clauses are implicitly universally closed. The method of this paper can be extended the the more general case of the language presented in [5], where universal and existential quantifiers occur explicitely. In this case there can be shared variables between a clause and the enclosing environment, as for instance in $\forall X \Box(p(X) :- [a](\forall Y q(X,Y) \supset r(X)))$ where variable X in q is the same as the one in r and p.[5]

This case can be dealt with by extending the logic so that a modal operator name can be any term, instead of a constant. Then, the first step of the translation will yield

$$\Box[a(X)]^c q(X,Y)$$
$$\Box(p(X) :- [a(X)]^c r(X))$$

so that the value of the shared variable will be passed through the name of the new modal operator.

Note that the approach of this paper provides a way of translating logic programming languages extended with embedded implication (such as N-Prolog) into standard Prolog. In fact, the semantics of those languages is given by intuitionistic logic, and it is well known that intuitionistic logic can be translated into modal logic $S4$. Since our $\Box$ operator is $S4$, by translating an N-Prolog clause into modal logic, we obtain a clause in our language, which can be translated to Prolog with the technique presented in this paper.

A similar approach applies as well to other languages with embedded implication such as CondLP [11], a language which supports hypothetical updates

[5] This way of dealing with variables has also strong similarities with the compilation scheme proposed in [15] for a first-order version of hereditary Harrop formulas.

together with integrity constraints, and makes use of a revision mechanism to restore consistency when an update violates some integrity constraint.

References

1. M. Abadi and Z. Manna. Temporal logic programming. *J. Symbolic Computation*, (8):277–295, 1989.
2. Y. Auffray and P. Enjalbert. Modal theorem proving: An equational viewpoint. *Journal of Logic and Computation*, 2(3):247–297, 1992.
3. M. Baldoni, L. Giordano, and A. Martelli. A multimodal logic to define modules in logic programming. In *Proc. 1993 International Logic Programming Symposium*, pages 473–487, Vancouver, 1993.
4. M. Baldoni, L. Giordano, and A. Martelli. A modal extention of logic programming: modularity, beliefs and hypothetical reasoning. Technical report, Dipartimento di Informatica, University of Turin, 1995.
5. M. Baldoni, L. Giordano, and A. Martelli. A modal extention of logic programming. In *Proc. 1994 Joint Conference on Declarative Programming GULP-PRODE 1994*, volume 2, pages 324–335, Peñiscola, Spain, September, 1994.
6. F. Debart, P. Enjalbert, and M. Lescot. Multimodal logic programming using equational and order-sorter logic. *Theoretical Computer Science*, (105):141–166, 1992.
7. L. Fariñas del Cerro. Molog: A system that extends Prolog with modal logic. *New Generation Computing*, (4):35–50, 1986.
8. L. Fariñas del Cerro and A. Herzig. Modal deduction with applications in epistemic and temporal logics. In D. M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 4, pages 499–594. Oxford Science Publications, 1995.
9. M. Fitting. Basic modal logic. In D. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic programming*, volume 1, pages 365–448. Oxford Science Publications, 1993.
10. D. M. Gabbay. Elements of algoritmic proof. In T.S.E. Maibaum S. Abramsky, D.M. Gabbay, editor, *Handbook of Logic in Theoretical Computer Science*. Oxford University Press, 1992.
11. D. M. Gabbay, L. Giordano, A. Martelli, and N. Olivetti. Hypothetical updates, priority and inconsistency in a logic programming language. In *Proc. of the Third Int. Conference on Logic Programming and Non-monotonic Reasoning*, volume 928 of *LNAI*, pages 203–216, Lexington, KY, USA, 1995. Springer Verlag.
12. M. Genesereth and N. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, 1987.
13. L. Giordano and A. Martelli. Structuring logic programs: a modal approach. *Journal of Logic Programming*, 21:59–94, 1994.
14. J. Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.
15. B. Jayaraman, K. Know, and G. Nadathur. Scoping constructs in logic programming: Implementation problems and their solution. Technical Report No. CS-1994-35, Duke University, October, 1994. To appear in Journal of Logic Programming, November 1995.
16. D. Miller. A logical analysis of modules in logic programming. *Journal of Logic Programming*, (6):79–108, 1989.
17. A. Nonnengard. How to use modalities and sorts in Prolog. In *Proc. of the JELIA'94: Logics in Artificial Intelligence*, volume 838 of *LNAI*, pages 365–378, York, UK, 1994. Springer Verlag.
18. H. J. Ohlbach. Semantics-based translation method for modal logics. *Journal of Logic and Computation*, 1(5):691–746, 1991.
19. D. S. Warren. Database updates in pure Prolog. In *Proc. Int. Conf. on Fifth Generation Computer Systems*, pages 244–253, Tokyo, 1984.

This article was processed using the LaTeX macro package with LLNCS style