A Framework for
Modal Logic Programming

Matteo Baldoni, Laura Giordano and Alberto Martelli
Dipartimento di Informatica — Università degli Studi di Torino
C.so Svizzera, 185 — I-10149 Torino, ITALY
{baldoni,laura,mrt}@di.unito.it

Abstract

In this paper we present a framework for developing modal extensions of logic programming, which are parametric with respect to the properties chosen for the modalities and which allow sequences of modalities of the form \([t_i] \ldots [t_n] \alpha \supset [s_1] \ldots [s_m] \alpha\). The language can deal with many of the well-known modal systems and several examples are provided. Due to its features, it is particularly suitable for performing epistemic reasoning, defining parametric and nested modules, describing inheritance in a hierarchy of classes and reasoning about actions.

A goal directed proof procedure of the language is presented, which is modular with respect to the properties of modalities. Moreover, we define a fixpoint semantics, by generalizing the standard construction for Horn clauses, which is used to prove soundness and completeness of the operational semantics with respect to model theoretic semantics, and it works for the whole class of logics identified by the inclusion axioms.

1 Introduction

Modal and temporal extensions of logic programming have recently received a lot of attention [20, 9, 1, 8, 15, 18], since they provide tools for formalizing temporal and epistemic knowledge and reasoning, while retaining the characterizing properties of logic programming languages, as, for instance, goal directed proof procedure, fixpoint semantics and the notion of minimal Herbrand model.

In this paper we present a framework for developing modal extensions of logic programming which are parametric with respect to the properties of modalities. The modalities are of the form \([t]\), where \(t\) is an arbitrary term, and they may occur in front of clauses, clause heads and in front of goals; their properties are specified by a set \(A\) of characterizing axioms. In particular we will focus on a class of normal modal logics characterized by axioms of the form

\[ [t_1] \ldots [t_n] \alpha \supset [s_1] \ldots [s_m] \alpha. \]  (1)

The logics in this class have been called inclusion logics in [14], where a refinement of the functional translation method [19] has been proposed for them.

Given a set of inclusion axioms \(A\) of the form (1), we define a modal logic programming language, called \(M_A\), whose underlying modal logic is the one characterized by those axioms. A goal directed proof procedure which is modular with
respect to the chosen set of axioms \( A \) is presented, by making use of a notion of matching relation between sequences of modalities, which only depends on the properties of modalities themselves. More specifically, the proof procedure is based on a notion of modal context. A modal context is a sequence of modal operators which keeps trace of the ordering between modalities found in front of goals during a computation, so that a modal context is associated with each goal to be solved. The matching relation is used to select a clause for proving a goal in a certain modal context, according to the properties of its modalities.

We show that our goal directed proof procedure for the languages \( \mathcal{MA} \)'s is sound and complete with respect to the model theoretic semantics for the modal logic characterized by the set of axioms \( A \). To do this we define a fixpoint semantics for the \( \mathcal{MA} \)'s, by generalizing the standard construction for Horn clauses, and we prove its completeness with respect to the model theoretic semantics through a canonical model construction. Though the construction is pretty standard, we believe that its merit is in the modularity of the approach, i.e., both the completeness and soundness proof are modular with respect to the set of chosen axioms \( A \), they work for the whole class of modal languages identified by the inclusion axioms.

This framework allows to define modal languages which extend the one proposed in [4] for representing module constructs. In particular, we show an example of how to deal with nested and parametric modules. Moreover, the framework allows to define languages suitable for performing epistemic reasoning, a simple form of reasoning about actions, and for interpreting some features of object-oriented paradigms in logic programming, such as hierarchical dependencies and inheritance among classes.

This paper is organized as follows. The class of languages \( \mathcal{MA} \)'s is introduced in Section 2 and some examples are discussed. In Section 3 we present the Kripke semantics for a class of first order multimodal languages of which the languages in this framework are clausal fragments. The operational semantics is discussed in Section 4, and some examples of derivation are shown. Finally, in Section 5, we define the fixpoint semantics and we sketch out the proof of soundness and completeness of operational semantics with respect to model theoretic semantics.

2 The syntax and some examples

In this section we introduce the class of modal logic programming languages \( \mathcal{MA} \)'s, which belongs to our framework, and we give some examples of programs.

Now we introduce the syntax of clauses and goals, which does not depend on the set of axioms \( A \). It extends Horn clause logic allowing modalities to occur in clauses and in goals. In particular, it allows free occurrences of some universal modalities of the form \([t]\), where \( t \) is an arbitrary term of the language, in front of clauses, clause heads and goals.

Let \( \mathcal{L} \) be an atomic formula of the form \( p(t_1, \ldots, t_r) \), with \( p \) a predicate symbol and \( t_1, \ldots, t_r \) terms of the language, \( T \) a distinguished symbol \( (true) \), and let \( t \) be an arbitrary term of the language. The abstract syntax is the following:

\[
G \ ::= \ T \ | \ A \ | \ G_1 \land G_2 \ | \ \exists x \ G \ | \ [t]G \\
D \ ::= \ G \ | \ H \ | \ [t]D \ | \ \forall x D \\
H \ ::= \ A \ | \ [t]H
\]

where \( G \) stands for a goal, \( D \) for a clause, \( H \) for a clause head\(^1\).

\(^1\)Note that, though we have explicitly introduced universal quantifiers in front of clauses and
A program $P$ consists of a set of closed clauses $D$, which have the general form
$\Gamma_b(G \supset \Gamma_h A)$, where $\Gamma_b$ and $\Gamma_h$ are arbitrary sequences of modalities (including the empty one).

In the following, we give some examples of programs. As we have mentioned in the introduction, the framework is based on a normal modal logic\textsuperscript{2}, and it is parameteric with respect to the logical properties of the modalities, which are specified through a set $\mathcal{A}$ of characterizing inclusion axioms of the form (1). Therefore, we will need to define $\mathcal{A}$ for each example in order to select a modal logic and to specify the language $\mathcal{M}_A$.

**Example 2.1 (Epistemic reasoning: The friends puzzle)** Let us consider the following situation. Peter is a friend of John, so if Peter knows that John knows something then John knows that Peter knows the same thing. That is, we assume the persistence axiom $P(p, j): [p][j] \alpha \supset [j][p] \alpha$, where $[p]$ and $[j]$ are modalities of type S4 (or $KT4$):

\[
\begin{align*}
T(p): [p] \alpha & \supset \alpha \\
4(p): [p] \alpha & \supset [p][p] \alpha
\end{align*}
\]

and $[p] \alpha$ and $[j] \alpha$ mean Peter knows $\alpha$ and John knows $\alpha$, respectively. Peter is married, so if Peter’s wife knows something, then Peter knows the same thing, that
is the axiom $I(w(p), p): [w(p)] \alpha \supset [p] \alpha$ holds, where $[w(p)]$ is a modality of type S4 representing the knowledge of Peter’s wife. John and Peter have an appointment. Let us consider the following situation:

(1) $[p]time$.  (3) $[w(p)][p]time \supset [j]time$.

(2) $[p][j]place$.  (4) $[p][j](place \land time \supset appointment)$.

That is, (1) Peter knows the time of their appointment; (2) Peter also knows that John knows the place of their appointment. Moreover, (3) Peter’s wife knows that if Peter knows the time of their appointment, then John knows that too (since John and Peter are friends); and finally (4) Peter knows that if John knows the place and the time of their appointment, then John knows that he has an appointment.

We can ask if each of the two friends knows that the other one knows that he has an appointment, by the following goal $G = [j][p]appointment \land [p][j]appointment$, which succeeds from the clauses above.

**Example 2.2 (Reasoning about actions: The shooting problem)** Assume that our language contains a $K$ modality $[a]$ for each possible atomic action $a$, and modalities $[a_1; a_2]$ to represent sequences of actions and a modality $[e]$ to represent the initial state. The set $\mathcal{A}$ will contain the logical axioms $[a_1][a_2] \alpha \supset [a_1; a_2] \alpha$, for all action sequences $a_1$ and $a_2$. We formalize the well known “shooting problem”, with the following set of clauses:

(1) $[e]alive$.  (5) $[S](alive \land A \neq shoot \supset [A]alive)$.

(2) $[e]unloaded$.  (6) $[S](alive \land unloaded \supset [shoot]alive)$.

(3) $[S](loaded \supset [shoot]dead)$.  (7) $[S](loaded \land A \neq shoot \supset [A]loaded)$.

(4) $[S; load]loaded$.  (8) $[S](unloaded \land A \neq load \supset [A]unloaded)$.

\textit{existential quantifiers in front of goals, as for Horn clauses, they could have been omitted and left implicit.}

\textsuperscript{2}As we will see below, for all modalities we will assume the axiom schema $K(t): [t][\alpha \supset \beta] \supset ([t][\alpha] \supset [t][\beta])$.}
where the clauses (1) and (2) represent the initial facts, the clauses (3) and (4) the causal rules, and the clauses (5)-(8) the frame axioms. In this example it is worth using modalities labelled with terms which contains variables to represent abstract sequences of actions. The goal \( G = [\varepsilon; \text{load}; \text{wait}; \text{shoot} \downarrow \text{dead} \text{ succeeds.} \) Moreover, the goal \( G' = [Z] \text{dead} \text{succeeds with } Z = \varepsilon; \text{load}; \text{shoot.} \)

Example 2.3 (Parametric and nested modules) In [4] a multimodal logic language with modules was presented. There, some modalities \([m_i] \) of type \( K \) were used for representing what is true in a module (each \( m_i \) can be regarded as a module name). This provides a simple way to define a flat collection of modules and to specify the proof of a goal in a module. In particular, a module \( m_i \) is defined through a set of definitions of the form \([\text{any}] [m_i] D\), by which each clause \( D \) is declared to belong to the module. The modality \([\text{any}] \) of type \( KT4 \) in front of the module definition is needed to make the definition visible in any context (and, in particular, from inside other modules). To this purpose the following inclusion axiom \( I(\text{any}, m_i); [\text{any}] \alpha \supset [m_i] \alpha \) is required. To prove a goal \( G \) in a module \( m_i \), we simply have to ask the goal \([m_i] G\).

In our framework we can define a language which extends the language in [4], by allowing an arbitrary sequence of modal operators in front of clauses. We can generalize module definitions \([\text{any}] [m_i] D\) above to nested module definitions as \([\text{any}] [m_i] [\text{any}] [m_j] D\), where the module \( m_j \) is defined locally to \( m_i \), and it visible only whenever \( m_i \) is entered. Finally, the parametric module definitions are achieved by sharing some variables between the label of the modalities (the name of a module) and their associated clauses (the body of a module).

Let us consider the following module definition (for readability we put module name in front of the sequence of clauses of the module, rather than in front of each one):

\[
\begin{align*}
[\text{any}] [\text{lists}] &\{ \\
& \quad \text{append}([], X, X), \\
& \quad \text{append}(Y, Z, Y1) \supset \text{append}([X|Y], Z, [X|Y1]). \ldots \\
[\text{any}] [\text{sort(Order)}] &\{ \\
& \quad [\text{any}] [\text{ascending}]\{ \\
& \quad \quad X < Y \supset \text{orded}(X, Y). \ldots \} \\
& \quad [\text{any}] [\text{descending}]\{ \\
& \quad \quad X > Y \supset \text{orded}(X, Y). \ldots \} \\
[\text{lists}] &\text{append}(X, [A, B|Y], L) \land [\text{Order}] \text{orded}(B, A) \land \\
& \quad [\text{lists}] \text{append}(X, [B, A|Y], M) \land \text{busort}(M, S) \supset \text{busort}(L, S). \\
& \text{busort}(S, S). \ldots \\
\end{align*}
\]

The module \( \text{lists} \) contains the definition of \( \text{append} \) and the other predicates on \( \text{lists}, \) while the module \( \text{sort(Order)} \) contains the definition of the predicate \( \text{busort} \) for ordering a list according to the bubblesort algorithm. In order to parametrize the algorithm with respect to the type of the order, we introduce within the parametric module \( \text{sort(Order)} \) two local modules, named \( \text{ascending} \) and \( \text{descending}, \) which contain two different definition of the predicate \( \text{orded}. \) Now, we can specify a particular order through the variable \( \text{Order}. \) Thus, the goal \( G_1 = [\text{sort(ascending)}] \text{busort}([2, 3, 1], S) \) succeeds with answer \( S = [1, 2, 3], \) while the goal \( G_2 = [\text{sort(descending)}] \text{busort}([2, 3, 1], S) \) succeeds with answer \( S = [3, 2, 1]. \)

Example 2.4 (Inheritance and hierarchy) The following example is taken from [7] and describes inheritance in a hierarchy of classes. Let us consider four classes,
named, respectively, animal, horse, bird and tweety. Since what is true for animals is also true for birds and horses, the bird and horse class inherit from the animal class. Moreover, the class tweety inherits from bird and thus from animal. To model this situation, we use four modal operators \([\text{animal}], [\text{horse}], [\text{bird}]\) and \([\text{tweety}]\), for describing the features of those classes; and the following set of inclusion axioms for defining the inheritance rules:

\[
\begin{align*}
I(\text{class}(\text{animal}), \text{class}(\text{horse})) & : \text{class}(\text{animal}) \supset \text{class}(\text{horse}) \alpha \\
I(\text{class}(\text{animal}), \text{class}(\text{bird})) & : \text{class}(\text{animal}) \supset \text{class}(\text{bird}) \alpha \\
I(\text{class}(\text{bird}), \text{class}(\text{tweety})) & : \text{class}(\text{bird}) \supset \text{class}(\text{tweety}) \alpha
\end{align*}
\]

while the classes are so described:

\[
\begin{align*}
[\text{class(\text{animal})}] & \{ \\
\text{mode(walk)} & , \\
\text{no.of legs}(X) \land X \geq 2 & \supset \text{mode(run)} , \\
\text{no.of legs}(X) \land X = 4 & \supset \text{mode(gallop)} \ldots \}
\end{align*}
\]

\[
\begin{align*}
[\text{class(\text{horse})}] & \{ \\
\text{no.of legs}(4) & , \\
\text{covering}(\text{hair}) \ldots \}
\end{align*}
\]

\[
\begin{align*}
[\text{class(\text{bird})}] & \{ \\
\text{no.of legs}(2) & , \\
\text{covering}(\text{feather}) & , \\
\text{mode}(\text{fly}) \ldots \}
\end{align*}
\]

\[
\begin{align*}
[\text{class(\text{tweety})}] & \{ \\
\text{owner}(\text{fred}) \ldots \}
\end{align*}
\]

The goal \(G_1 = [\text{class(\text{bird})}] \text{mode(\text{run})}\) succeeds, since the clause defining \(\text{mode(\text{run})}\) is inherited by the class \(\text{bird}\) from \(\text{animal}\). Moreover, the goal \(G_2 = [\text{class}(X)] \text{mode(\text{fly})}\) succeeds too, with \(X = \text{bird}\) or \(X = \text{tweety}\).

### 3 Model theoretic semantics

In this section we introduce the class of modal logics on which the class of languages presented in the previous section are based, by defining a Kripke semantics for them.

First, let us define a first order multimodal language \(\mathcal{L}_{A,[\theta]}\), containing the logical connectives \(\neg, \land, \supset\), quantifiers \(\forall\) and \(\exists\), and the modal operators \([\theta]\). We assume the language contains countably many variables, constants, function symbols and relation symbols. According to the choice of the logical axioms \(A\), different modal logics can be obtained. We only consider normal modal logics, hence, we always assume that axiom \(K(\theta)\): \([\theta](\alpha \supset \beta) \supset ([\theta]\alpha \supset [\theta]\beta)\) holds, for all modalities. We focus on a particular class of normal modal logics, those that in [14] are called inclusion logics which are characterized by logical axioms of the form (1). Indeed, these axioms determine inclusion properties on the accessibility relations.

A logic programming language \(\mathcal{M}_{A}\) is a clausal fragment of \(\mathcal{L}_{A,[\theta]}\), and its underlying logic is the one characterized by the axioms in \(A\).

In a first order Kripke interpretation each world is associated with a domain of quantification. We will not assume that domains are constant. The only restriction we put on them is that the domain of a world \(w\) is contained in the domain of all worlds reachable from \(w\), i.e., domains are increasing (or monotone). In each interpretation we will fix a nonempty set \(D\) of possible objects. The domain of each world will be a subset of \(D\).

We define a Kripke interpretation for the language \(\mathcal{L}_{A,[\theta]}\), and we call it a Kripke \(A\)-interpretation.

**Definition 3.1** A Kripke \(A\)-interpretation is an ordered tuple \(M = \langle W, R, D, \mathcal{D}, V \rangle\), where:
• $W$ is a nonempty set of worlds;
• $D$ is a nonempty set of objects;
• $D$ is a function from $W$ to (non-empty) subsets of $D$ (it associates a domain with each world), satisfying the following condition: for all $w, w' \in W$, if $(w, w') \in \mathcal{R}^3$ then $D(w) \subseteq D(w')$;
• $V$ is an assignment function, such that:
  - for each variable $x \in \mathcal{L}_A[t]$, $V(x) \in D$;
  - for each $n$-ary function symbol $f$ of $\mathcal{L}_A[m]$ (including constants of $\mathcal{L}_A[t]$), $V(f) \in D^n \to D$;
  - for each $n$-ary predicate symbol $p$ and each world $w \in W$, $V(p, w) \subseteq D^n$,
    i.e., $V(p, w)$ is a set of $n$-tuples $\langle a_1, \ldots, a_n \rangle$, where each $a_i$ is an element in $D$;
• $\mathcal{R}$ is the accessibility relation. It is parametrized with respect to domain elements, i.e. for each domain element $d \in D$ the accessibility relation $\mathcal{R}_d$ is a binary relation on $W$; moreover, for each axiom schema $[t_1] \ldots [t_n][\alpha] \supset [s_1] \ldots [s_m][\alpha]$ in $A$, we have that $\mathcal{R}_V(t_1) \circ \ldots \circ \mathcal{R}_V(t_n) \supset \mathcal{R}_V(s_1) \circ \ldots \circ \mathcal{R}_V(s_m)$;
  where $\mathcal{R}_V(t) \circ \mathcal{R}_V(t') = \{(w, w') : (w, w') \in \mathcal{R}_V(t) \text{ and } (w', w'') \in \mathcal{R}_V(t')\}$.

Interpretation for terms in the domain is defined as usual from the interpretation of constants and function symbols.

Let $M$ be a Kripke $A$-interpretation, let $w \in W$ be a world, and let $V$ be an assignment function. Then, we say that a formula $\alpha$ of $\mathcal{L}_A[t]$ is satisfied by $V$ in the Kripke $A$-interpretation $M$, denoted by $M, w \models A \alpha$ at $w$, if the following conditions hold:
• $M, w \models A T$;
• $M, w \models A p(t_1, \ldots, t_n) \iff (V(t_1), \ldots, V(t_n)) \in V(p, w)$;
• $M, w \models A \neg \alpha \iff M, w \not\models A \alpha$;
• $M, w \models A \alpha \land \beta \iff M, w \models A \alpha$ and $M, w \models A \beta$;
• $M, w \models A \alpha \lor \beta \iff M, w \not\models A \alpha$ or $M, w \models A \beta$;
• $M, w \models A \forall x \alpha \iff$ for every variable assignment $V'$ that agrees with $V$ everywhere except on $x$, and such that $V'(x) \in D(w)$, $M, w \models A' \alpha$;
• $M, w \models A \exists x \alpha \iff$ for some variable assignment $V'$ that agrees with $V$ everywhere except on $x$, and such that $V'(x) \in D(w)$, $M, w \models A' \alpha$;
• $M, w \models A [t] \alpha \iff$ for all $w' \in W$ such that $(w, w') \in R_V(t)$, $M, w' \models A \alpha$.

A formula $\alpha$ of the language $\mathcal{L}_A[t]$ is satisfiable if there is a Kripke $A$-interpretation $M = \langle W, \mathcal{R}, D, V \rangle$ and some $w \in W$ with every term of $\alpha$ interpreted in $D(w)$ such that $M, w \models A \alpha$. We say that a formula $\alpha$ is valid, written $\models A \alpha$, if for every Kripke $A$-interpretation $M = \langle W, \mathcal{R}, D, V \rangle$, for every $w \in W$ with every term of $\alpha$ interpreted in $D(w)$, $M, w \models A \alpha$.

---

$^3$That is, if there exists a parameter $d \in D$ such that $(w, w') \in \mathcal{R}_d$. 
Notice that, since the domain may change from a world to another, there is the problem of defining the satisfiability at a world \( w \) of a formula \( \alpha(t) \) containing a term \( t \) whose interpretation is not in \( D(w) \). With this regard we follow [11], and we do not make any special restriction, like imposing \( \alpha(t) \) to be false in \( w \), or to be undefined. However, when we define satisfiability and validity of a formula we look at the truth value of the formula in an interpretation at a certain world only if the interpretation of each term in the formula is in the domain of that world.

In general, when function symbols are present, each function symbol can be given a different interpretation at each different world. In the Kripke semantics above, however, function symbols are given the same interpretation in all possible worlds. As a consequence, closed terms have the same interpretation in all possible worlds (rigid designators). On the contrary, predicate symbols may have a different interpretation at each possible world. For a survey of the different systems for quantified modal logic see [13].

4 Operational semantics

Now that we have defined the class of modal logics underlying the programming languages \( \mathcal{M}_A \)'s, we are ready to introduce a goal directed proof procedure for them. The proof procedure will be modular with respect to the logical axioms \( A \).

Since modalities are allowed to occur freely in front of goals, when proving a goal \( G \) from a program \( P \) we need to record the sequence of modalities which occur in the goal, that is the modal context in which each subgoal has to be proved. According to the modal context in which a subgoal has to be proved, a given clause of the program may be used or not to solve it: it depends on the modal structure of the clause itself, and on its relation to the modal context of the goal. For instance, given a goal \( [t_1][t_2]p \), the sequence \( [t_1][t_2] \) represents the modal context for the goal \( p \). Assume that the program contains a clause \( [t_3]p \). This clause can be used to solve the goal \( p \) only if the modality \( [t_3] \) relates somehow to the context \( [t_1][t_2] \). For instance, if our set \( A \) of logical axioms contains the axiom schema \( [t_3] \alpha \supset [t_1][t_2] \alpha \), then the clause can certainly be used to prove the goal.

We formalize this relationship between sequences of modalities (the modalities in the clause and the modalities in the modal context of a goal) by introducing a matching relation between them. This relation will depend on the logical axioms \( A \) of the logic. Given two sequences of modalities \( \Gamma_1 \) and \( \Gamma_2 \), we will write \( \Gamma_1 \trianglelefteq A \Gamma_2 \), to mean that \( \Gamma_1 \) matches \( \Gamma_2 \) in the logic characterized by axioms \( A \).

In the following we define a goal directed proof procedure which is modular with respect to the axioms \( A \) of the logic: the differences among the logics are factored out in the adopted matching relation \( \trianglelefteq A \).

The proof procedure we define is an abstract one. In particular, we follow [17], in order to avoid problems with variable renaming and substitutions, we denote by \( [P] \) the set of all ground instances of a given set of clauses in \( P \).

**Definition 4.1** Let be \( \Theta \) a set of clauses and \( \Gamma \) an arbitrary modal context. Define \( [\Theta] \) to be the smallest set of formulas satisfying the following conditions:

- \( \Theta \subseteq [\Theta] \);
- if \( \forall \alpha \in [\Theta] \) then \( \Gamma(D'[t/x]) \in [\Theta] \) for all ground terms \( t \).

\(^4\)Observe that the ground term \( t \) replaces also the variables belonging to the terms of the modalities.
Hence, given a program $P$, $[P]$ contains ground clauses of the form $\Gamma_b(G \supset \Gamma_b A)$. The operational derivability of a closed goal $G$ from a program $P$ in a modal context $\Gamma$, is defined by induction on the structure of $G$. We introduce a proof rule for each kind of goal. Moreover, we denote by $\Gamma_1 | \Gamma_2$ the concatenation of the modal contexts $\Gamma_1$ and $\Gamma_2$.

**Definition 4.2 (Operational Semantics)** Given a set of axioms $A$, a program $P$, and a modal context $\Gamma$, operational derivability of a goal $G$ from $P$ in the modal context $\Gamma$, that is $\langle P, \Gamma \rangle \vdash_A G$, is defined by induction on the structure of $G$ as follows:

1. $\langle P, \Gamma \rangle \vdash_A T$;
2. $\langle P, \Gamma \rangle \vdash_A A$ if there is a clause $\Gamma_b(G \supset \Gamma_b A) \in [P]$ and $\Gamma_b \preceq_A \Gamma_b$, and $\langle P, \Gamma_b \rangle \vdash_A G$;
3. $\langle P, \Gamma \rangle \vdash_A G_1 \land G_2$ if $\langle P, \Gamma \rangle \vdash_A G_1$ and $\langle P, \Gamma \rangle \vdash_A G_2$;
4. $\langle P, \Gamma \rangle \vdash_A [t]G$ if $\langle P, \Gamma[[t]] \rangle \vdash_A G$;
5. $\langle P, \Gamma \rangle \vdash_A \exists x \; G$ if $\langle P, \Gamma \rangle \vdash_A G[t/x]$, for some ground term $t$.

Proving a goal $G$ from a program $P$ amounts to show that $G$ is operationally derivable from $P$ in the empty modal context $\varepsilon$, that is, to show that $\langle P, \varepsilon \rangle \vdash_A G$ can be derived by making use of the above proof rules.

While inference rules 1), 3) and 5) are the usual ones for dealing with distinguished proposition $T$, conjunctive goals and existential goals, rules 2) and 4) are those which deal with modalities. By rule 4), to prove a goal $[t]G$, the modality $[t]$ is added to the current context $\Gamma$, and the goal $G$ is proved for the newly extended context $\Gamma[[t]]$. By rule 2), a clause $\Gamma_b(G \supset \Gamma_b A)$ can be selected from $P$ to prove an atomic formula $A$ in a given context $\Gamma$, if the modalities occurring in front of the clause and in front of the clause head are in a certain relation with $\Gamma$, if $\Gamma_b$ and $\Gamma_b$ match $\Gamma$ according to the properties of modalities specified by the set of axioms $A$.

More formally, let $\mathcal{C}$ be a set of all ground modalities of the form $[t]$, where $t$ is a ground term of the language $M_A$. We define the set of modal contexts $\mathcal{C}^*$ as the set of all finite sequences on $\mathcal{C}$, including the empty sequence “$\varepsilon$”.

**Definition 4.3 (Matching relation $\preceq_A$)** Given a set $A$ of axioms, the matching relation $\preceq_A$ is the least relation on $\mathcal{C}^*$ which satisfies the following properties:

- $\forall \Gamma, \Gamma' \in \mathcal{C}^*$, if $\Gamma \alpha \supset \Gamma' \alpha \in A$, then $\Gamma \preceq_A \Gamma'$;
- $\forall \Gamma \in \mathcal{C}^*$, $\Gamma \preceq_A \Gamma$ (reflexivity);
- $\forall \Gamma_1, \Gamma_2, \Gamma_3 \in \mathcal{C}^*$, if $\Gamma_1 \preceq_A \Gamma_2$ and $\Gamma_2 \preceq_A \Gamma_3$ then $\Gamma_1 \preceq_A \Gamma_3$ (transitivity);
- $\forall \Gamma_1, \Gamma_2, \Gamma^2 \in \mathcal{C}^*$, if $\Gamma_1 \preceq_A \Gamma_2$ then $\Gamma'[\Gamma_1] \preceq_A \Gamma'[\Gamma_2]$ (isotonicity).

From the above definition of $\preceq_A$ we can prove the following property.

**Proposition 4.1** Given a set of axioms $A$, for all $\alpha \in \mathcal{L}_A[[t]]$ and for all $\Gamma, \Gamma' \in \mathcal{C}^*$, if $\Gamma \preceq_A \Gamma'$ then $\models_A \Gamma \alpha \supset \Gamma' \alpha$. 
Example 4.1 (The friends puzzle) From axioms of the Example 2.1, we get the following set of relations \([p] \succeq_A [p], [j] \succeq_A [j], [p][p] \succeq_A [p][p], [w(p)] \succeq_A [w(p)][w(p)], [w(p)] \succeq_A [p][p]\). The matching relation \(\succeq_A\) is defined as the reflexive, transitive and isotonic closure of the set of relations above. The goal \([j][p]\) appointment \(\wedge [p][j]\) appointment succeeds. We show the derivation of the first conjunct \([j][p]\) appointment.

1. \(\varepsilon \vdash_A [j][p]\) appointment
2. \( [j][p] \vdash_A \text{place} \wedge \text{time} \) by clause (4) and \([p][j] \succeq_A [j][p]\),
3. \( [j][p] \vdash_A \text{place} \)
4. \( [j][p] \vdash_A \text{time} \)
5. success, by clause (2) and \([p][j] \succeq_A [j][p]\),
6. \( [j][p] \vdash_A \text{time} \)
7. \( [p][p] \vdash_A \text{time} \) by clause (3) and \([w(p)] \succeq_A [p], [p][j] \succeq_A [j][p]\),
8. \( [p][p] \vdash_A \text{time} \)
9. success, by clause (1) and \([p] \succeq_A [p][p]\).

Note that, when the language contains only axioms of the form \([k_1] \alpha \supset [k_2] \alpha\) in which there is a single modality on the antecedent, the proof procedure can be simplified. In particular, due to the specificity of the matching conditions, proof rule 2) for atomic formulas can be simplified as follows:

2'. \( \langle P, \Gamma \rangle \vdash_A A \) if there is a clause \( \Gamma_b(G \supset \Gamma_{b}(A)) \in [P] \) such that, for some \( \Gamma^*_{\alpha} \) and \( \Gamma^*_{\beta} \), \( \Gamma^*_{\alpha} \Gamma^*_{\beta} = \Gamma \), \( \Gamma_b \succeq_A \Gamma^*_b \), \( \Gamma_b \succeq_A \Gamma^*_b \), and \( \langle P, \Gamma^*_b \rangle \vdash_A G \).

This is the kind of semantics we have used in [4], where a modal logic programming language is proposed to define modularity constructs, and where modalities were ruled by the axioms of S4 and K. In the general case, this is not sufficient, and we must require that \( \Gamma_b \) and \( \Gamma_b \) jointly match the current context \( \Gamma \). An example is given by the derivation above, where 7. is obtained from 6. and clause (3), by applying rule 2), while it could not be obtained by applying rule 2'.

Example 4.2 (The shooting problem) From axioms of the Example 2.2, we get the relation \([s_1][s_2] \succeq_A [s_1]; [s_2] \) for all action sequences \( s_1 \) and \( s_2 \). The goal \( G = [\varepsilon; \text{load}; \text{wait}; \text{shoot}; \text{dead}] \) succeeds with the following derivation.

1. \( \varepsilon \vdash_A [\varepsilon; \text{load}; \text{wait}; \text{shoot}; \text{dead} \)
2. \( [\varepsilon; \text{load}; \text{wait}; \text{shoot}] \vdash_A \text{dead} \)
3. \( [\varepsilon; \text{load}; \text{wait}] \vdash_A \text{loaded} \) by clause (3) and \( S = [\varepsilon; \text{load}; \text{wait}; \text{load}] \),
4. \( [\varepsilon; \text{load}; \text{load}] \vdash_A \text{loaded} \) by clause (7) and \( S = [\varepsilon; \text{load}; \text{load}] \),
5. success, by clause (4) and \( S = \varepsilon \) and \( [\varepsilon; \text{load}] \succeq_A [\varepsilon; \text{load}] \).

It is worth noting that the matching relation \( \succeq_A \) may be regarded as a derivation relation in a rewriting system \( R_A \), having as rewriting rules the pairs \((\Gamma_1, \Gamma_2)\) such that \( \Gamma_1 \alpha \supset \Gamma_2 \alpha \) belongs to \( A \). In fact, similar properties which are specified by Definition 4.3 can be used to define a derivation relation \( \Rightarrow^*_R \) between strings of \( \alpha^* \) generated by \( R_A \).

Therefore, we may say that, given two modal context \( \Gamma_1 \) and \( \Gamma_2 \), \( \Gamma_1 \succeq_A \Gamma_2 \) if and only if \( \Gamma_1 \Rightarrow^*_R \Gamma_2 \). In others words, to establish if \( \Gamma_1 \succeq_A \Gamma_2 \) is equivalent to
establish if \( \Gamma_2 \) can be derived from \( \Gamma_1 \) by means of a finite number of applications of the rewriting rules of \( R_A \). That is, to establish if \( \Gamma_2 \) belongs to the language 

\[ [\Gamma_1]_{R_A} = \{ \Gamma \in \mathcal{C}^* : \Gamma_1 \Rightarrow_{R_A} \Gamma \} . \]

This problem is known in literature as the word problem for the rewriting system. In general the word problem is undecidable\(^5\), nevertheless under certain restriction on such systems it is decidable. For example when the system is complete, i.e., it is noetherian and confluent \(^6\), or when \([\Gamma_1]_{R_A}\) is a context sensitive language\(^6\) \([16]\).

These remarks are quite relevant when we have to deal with the implementation of the matching relation in the case when only ground terms may occur within modalities in the program, in the goal and in the axioms \( A \), and, in particular, no variables may occur within them. In the general case, the problem of implementing the matching relation is more serious, and verifying if a sequence of modalities \( \Gamma_1 \) matches another sequence \( \Gamma_2 \) cannot be simply seen as the problem of determining if \( \Gamma_2 \) can be derived from \( \Gamma_1 \) by applying some rewriting rules. In fact, when the sequences \( \Gamma_1 \) and \( \Gamma_2 \) contain variables, and modalities in the axioms contain variables too, verifying if \( \Gamma_1 \) matches \( \Gamma_2 \) involves some form of theory unification.

## 5 Fixpoint semantics

In this section, we present a fixpoint semantics for the class of languages \( \mathcal{M}_A \)'s, which is used to prove soundness and completeness of the proof procedure in Section 4 with respect to the model theory defined in Section 3.

We define an immediate consequence operator \( T_{A,P} \) based on a relation of weak satisfiability for closed goals on the line of \([17]\). This allows to capture the dynamic evolution of the modal context in the operational semantics during a computation.

Completeness with respect to the model theory is proved by a Henkin-style canonical model construction, which is similar to the one given in [5]. We also show that there is no loss of generality in restricting Kripke \( A \)-interpretations to those in which the domain at each world is the Herbrand universe. It is worth noting that the \( T_{A,P} \) operator, canonical model construction, and all definitions and proofs are modular with respect to the chosen set of axiom \( A \). We think this is the merit of our approach. Proofs are omitted for lack of space.

The weak satisfiability is defined on a Kripke-like semantics, where each world represents a modal context and it interprets the program at that modal context. As a result we define an interpretation for a program \( P \) of a \( \mathcal{M}_A \) as any function \( I : \mathcal{C}^* \rightarrow 2^{B(P)} \); that is, a mapping from modal contexts to Herbrand interpretation of the program \( P \). We denote by \( \mathcal{I} \) the set of all interpretations. It is easy to note that \((\mathcal{I}, \subseteq_{\mathcal{E}})\) is a complete lattice, where \( \subseteq_{\mathcal{E}} \) is defined as the ordering \( I_1 \subseteq_{\mathcal{E}} I_2 \) if and only if \( (\forall \Gamma \in \mathcal{C}^*) I_1(\Gamma) \subseteq I_2(\Gamma) \). The bottom element, denoted by \( \bot \), is the interpretation such that \( \bot(\Gamma) = \emptyset \), for all context \( \Gamma \in \mathcal{C}^* \). Moreover, we define the join of two interpretations \( I_1 \) and \( I_2 \) as the interpretation \( (I_1 \cup I_2)(\Gamma) = I_1(\Gamma) \cup I_2(\Gamma) \), and the meet of \( I_1 \) and \( I_2 \) as the interpretation \( (I_1 \cap I_2)(\Gamma) = I_1(\Gamma) \cap I_2(\Gamma) \).

**Definition 5.1 (Weak satisfiability \( \models_{\mathcal{A}} \))** Let \( I \) be an interpretation and let \( \Gamma \) be a modal context, then we say that a closed goal \( G \) of \( \mathcal{M}_A \) is weakly satisfiable in \( I(\Gamma) \), denoted by \( I(\Gamma) \models_{\mathcal{A}} G \), by induction on the structure of \( G \) as follows:

1. \( I(\Gamma) \models_{\mathcal{A}} T \);

\(^5\)It can be reduced to the Post's Correspondence Problem.

\(^6\)In this case it is shown to be even a PSPACE-complete problem.
2. $I(\Gamma) \models_{\mathcal{A}} A$ iff $A \in I(\Gamma)$;

3. $I(\Gamma) \models_{\mathcal{A}} G_1 \land G_2$ iff $I(\Gamma) \models_{\mathcal{A}} G_1$ and $I(\Gamma) \models_{\mathcal{A}} G_2$;

4. $I(\Gamma) \models_{\mathcal{A}} \exists x G' \iff I(\Gamma) \models_{\mathcal{A}} G'[t/x]$, for some $t \in U_P$;

5. $I(\Gamma) \models_{\mathcal{A}} [t] G'$ iff $I(\Gamma') \models_{\mathcal{A}} G'$, for all $\Gamma' \in \mathcal{C}^*$ such that $\Gamma'[t] \models_{\mathcal{A}} \Gamma'$.

Given an interpretation $I$ and a context $\Gamma$, $I(\Gamma) \models_{\mathcal{A}} G$ means that the goal $G$ is true in the interpretation associated with $\Gamma$.

Note that in the rule 5) the matching relation $\models_{\mathcal{A}}$ between modal context depends on the choice of the set of axioms $\mathcal{A}$. A goal $[t] G$ holds in a world $\Gamma$ if the goal $G$ is true in all worlds reachable from $\Gamma$, that is in all world $\Gamma'$ such that $\Gamma'[t] \models_{\mathcal{A}} \Gamma'$. This allows to satisfy the inclusion relation properties of the Kripke $\mathcal{A}$-interpretation which is built by the fixpoint semantics (as we will see from the canonical model construction).

We are interested in finding an interpretation $I$ such that $G$ is operationally derivable from $P$ if and only if $I(\varepsilon) \models_{\mathcal{A}} G$. This particular interpretation is the least fixed point of the following immediate consequence transformation $T_{\mathcal{A}, P}$ defined in the domain of interpretations $(I, \sqsubseteq)$.

**Definition 5.2 (The function $T_{\mathcal{A}, P}$)** Let $P$ be a program in $\mathcal{M}_\mathcal{A}$, $\Gamma$ a modal context, and let $I$ be an interpretation, then we define a function $T_{\mathcal{A}, P}$ from interpretations to interpretations as follows

$$T_{\mathcal{A}, P}(I)(\Gamma) = \{ A \in B(P) : \Gamma_0(G \supset A) \in [P] \text{ and } \Gamma_0^*|_{\mathcal{A}} \models_{\mathcal{A}} \Gamma, \text{ for some } \Gamma^*_b \text{ such that } \Gamma^*_b \models_{\mathcal{A}} \Gamma^*_b, \text{ and } I(\Gamma^*_b) \models_{\mathcal{A}} G \}.$$ 

The transformation $T_{\mathcal{A}, P}$ is monotone and continuous in $(I, \sqsubseteq)$. Thus, the least fixed point $T^*_{\mathcal{A}, P}$ of $T_{\mathcal{A}, P}$ exists by monotonicity, and, by continuity, we have $T^*_{\mathcal{A}, P}(\bot) = \bigcup_{k \in \mathbb{N}} T^k_{\mathcal{A}, P}(\emptyset)$, where $T^0_{\mathcal{A}, P}(\emptyset) = \emptyset$, and for each $k > 0$, $T^k_{\mathcal{A}, P}(\emptyset) = T_{\mathcal{A}, P}(T^{k-1}_{\mathcal{A}, P}(\emptyset))$.

It is worth noting that for $T^*_{\mathcal{A}, P}(\bot)$ the following property holds, which is the fixpoint semantics counterpart of the inclusion relation property defined by the set $\mathcal{A}$ in the Kripke $\mathcal{A}$-interpretations.

**Proposition 5.1** Let $P$ be a program, $G$ a closed goal and let $\Gamma$ be a modal context, then $T^*_{\mathcal{A}, P}(\bot)(\Gamma) \models_{\mathcal{A}} G \iff T^*_{\mathcal{A}, P}(\bot)(\Gamma') \models_{\mathcal{A}} G$ for all context $\Gamma'$ such that $\Gamma \sqsubseteq_{\mathcal{A}} \Gamma'$.

The correctness of the fixpoint semantics with respect to the operational semantics is given by the following theorem.

**Theorem 5.1** (Soundness and Completeness w.r.t. operational semantics) Let $P$ be a program and let $G$ be a closed goal of $\mathcal{M}_\mathcal{A}$, then $(P, \varepsilon) \vdash_{\mathcal{A}} G \iff T^*_{\mathcal{A}, P}(\bot)(\varepsilon) \models_{\mathcal{A}} G$.

**Proof.** (If part) It is proved by showing, by induction on the derivation of $G$, the stronger property that, for any modal context $\Gamma$, if $(P, \Gamma) \vdash_{\mathcal{A}} G$ then $T^k_{\mathcal{A}, P}(\bot)(\Gamma) \models_{\mathcal{A}} G$. (Only if part) It is proved by showing, with a double induction on $k$ and the structure of $G$, that $T^k_{\mathcal{A}, P}(\bot)(\Gamma) \models_{\mathcal{A}} G$ implies $(P, \Gamma) \vdash_{\mathcal{A}} G$, for any modal context $\Gamma$ and $k \geq 0$. □
Let us now consider the completeness of the fixpoint semantics with respect to the model theory. The completeness proof is given by constructing a canonical model for a given program $P$, whose domain is constant and is the Herbrand universe $U_P$ of $P$.

**Definition 5.3 (Canonical Model)** The canonical model $M_{A,P}$ for a program $P$ in $M_A$ is a quintuple $(W, \mathcal{R}, D, \mathcal{D}, V)$, where:

- $W = \mathcal{C}^*$;
- $D = U_P$ (the Herbrand universe of $P$);
- $\mathcal{D}$ is the constant function $\mathcal{D}(w) = U_P$, for all $w \in W$;
- $V$ is an assignment function, such that:
  - (a) it interprets terms as usual in Herbrand interpretations;
  - (b) for each $n$-ary predicate symbol $p$ and each world $\Gamma \in \mathcal{C}^*$,
    
    $$V(p, \Gamma) = \{ \langle t_1, \ldots, t_n \rangle: T^\mathcal{A}_p(\bot)(\Gamma) \models_{A} p(t_1, \ldots, t_n) \text{ and } t_1, \ldots, t_n \in U_P \}.$$  

- $\mathcal{R}$ is defined as follows: for all $t \in U_P$,  

  $$\mathcal{R}_t = \{ (\Gamma, \Gamma') : \Gamma, \Gamma' \in \mathcal{C}^* \text{ and } \Gamma[t] \models_{A} \Gamma'[t] \}.$$ 

The canonical model $M_{A,P}$ for a program $P$ in $M_A$ is a Kripke $A$-interpretation. In fact, it is easy to see that for each axiom $[t_1] \ldots [t_n] \tau \supset [s_1] \ldots [s_m] \alpha$ in $A$, $\mathcal{R}_V(t_1) \circ \ldots \circ \mathcal{R}_V(t_n) \supset \mathcal{R}_V(s_1) \circ \ldots \circ \mathcal{R}_V(s_m)$ holds.

Completeness proof is based on the following two properties of $M_{A,P}$. They can be proved by induction on the structure of the goals $G$ and the clauses $D$.

**Theorem 5.2** Let $P$ be a program in $M_A$, $M_{A,P}$ its canonical model and let $G$ be a closed goal, then the following properties hold:

1. for any $\Gamma \in \mathcal{C}^*$, $M_{A,P}, \Gamma \models_{A} G \iff T^\mathcal{A}_p(\bot)(\Gamma) \models_{A} G$;
2. $M_{A,P}$ satisfies $P$; i.e., for all clauses $D$ in $P$, $M_{A,P}, \varepsilon \models_{A} D$.

By Theorem 5.2, the canonical model definition makes it explicit the fact that the fixed point construction builds a Kripke model for the program $P$. We can now prove the following result.

**Theorem 5.3 (Soundness and Completeness w.r.t. model theory)** Let $P$ be a program and let $G$ be a closed goal of $M_A$, then $\models_{A} P \supset G \iff T^\mathcal{A}_p(\bot)(\varepsilon) \models_{A} G$.

**Proof.** (If part) Let us assume that $\models_{A} P \supset G$. Then, for every Kripke $A$-interpretation $M = (W, \mathcal{R}, D, \mathcal{D}, V)$, for every $w \in W$, $M, w \models_{A} P \iff M, w \models_{A} G$. Hence, in particular for the canonical model $M_{A,P}$ and the world $\varepsilon \in \mathcal{C}^*$ it holds that $M_{A,P}, \varepsilon \models_{A} P \iff M_{A,P}, \varepsilon \models_{A} G$. By Theorem 5.2, property 2), we have that $M_{A,P}, \varepsilon \models_{A} P$, thus $M_{A,P}, \varepsilon \models_{A} G$ holds and then, by Theorem 5.2, property 1), $T^\mathcal{A}_p(\bot)(\varepsilon) \models_{A} G$. (Only if part) It can be proved by showing, with an easy double induction on iteration $k$ and on the structure of $G$, that $T^\mathcal{A}_p(\bot)(\Gamma) \models_{A} G \iff P \models_{A} \Gamma G$ holds for any modal context $\Gamma$ and $k \geq 0$. $\square$
As a corollary we have that given a program $P$ in the language $\mathcal{M}_A$, without loss of generality, we may consider only Kripke $A$-interpretations with constant domain $U_P$; that is, interpretations in which, for all worlds $w \in W$, $D(w) = U_P$.

**Corollary 5.1** Let $P$ be a program and $G$ a closed goal of $\mathcal{M}_A$, then $\models_A P \supset G \iff \models_{A,H} P \supset G$ where $\models_{A,H}$ denotes the satisfiability in Kripke $A$-interpretations with constant domain $U_P$.

## 6 Conclusions and related works

In this paper we have developed a framework for modal extensions of logic programming which is based on the class of inclusion logics. The modalities may occur in front of clauses, clause heads and in front of goals, and are of the form $[t]$, where $t$ is a term. Their properties are specified by a set of characterizing axioms $A$. We deal with axioms which determine inclusion properties between accessibility relations.

For this class of languages we have defined a goal directed proof procedure which is modular with respect to the properties of modalities: it makes use of a notion of matching between sequences of modalities, which depends on the properties of modalities themselves. Verifying that a sequence $\Gamma$ matches another one $\Gamma'$, in the case when no parametric modalities are allowed, essentially amounts to check if $\Gamma'$ can be derived from $\Gamma$ in some rewriting system (whose rewrite rules are determined by modal axioms). In the general case, some theory unification is required.

While, in languages of our framework, universal modalities are allowed to freely occur in front of clauses, clause heads and clause bodies (or goals), existential modal operators are not allowed. In particular, as a difference with other languages proposed in the literature, like TEMPOLOG [1], Temporal Prolog [12] and the language in [3], existential modalities are not allowed to occur in front of goals. In spite of this limitation, the features of parametric modalities and the possibility of introducing inclusion axioms, make the language well suited for performing some epistemic reasoning, for defining parametric and nested modules, for representing inheritance in a hierarchy of classes and for reasoning about action.

Actually, the class of languages could be extended to allow existential modalities in front of goals. Indeed, due to the analogy between universal (existential) quantifiers and universal (existential) modalities, and from the fact that, in standard logic programs, universal quantifiers occur in front of clauses, while existential quantifiers occur in front goals, the use of existential modalities should be possible. Of course, to deal with existential modalities $\langle t \rangle$ in front of goals, the proof procedure presented in Section 4 should be modified substantially. The main difference is that, since existential modalities $\langle t \rangle$ do not distribute on conjunctions, a goal $\langle t \rangle (G_1 \land G_2)$ cannot be proved by proving the two subgoals $\langle t \rangle G_1$ and $\langle t \rangle G_2$. For this reason, the policy of recording in a context $\Gamma$ the sequence of modalities that are found in front of a goal does not work in that case in a straightforward way.

The class of languages we have presented has strong similarities with MOLOG [9], which is an extension of Prolog with modal operators. In MOLOG modalities may occur in front of clauses, in front of clause heads, and in front of goals. MOLOG can be regarded as a framework, which can be instantiated with particular modal logics. In [9] a resolution procedure, close to Prolog resolution, is defined for modal Horn clauses in the logic $S5$ which contains only universal modal operators of the form $\text{Know}(a)$. Though the language in similar to ours, the properties of $S5$ modalities are different from the ones we have considered. In [3] a modal SLD-resolution method is presented for a fragment of MOLOG in which $\Box$ is disallowed.
in the bodies of modal clauses (while ◻ is allowed). Some different modal systems
(Q, T and K4) are considered. A fixpoint semantics is also provided.

Instead of developing specific theorem proving techniques for modal logics, many
authors have proposed the alternative approach of translating modal logics into
classical first order logic [10], so that standard theorem provers can be used. The
translation methods are based on the idea of making explicit reference to the worlds,
by adding to all predicates and functions as argument representing the world where
the predicate holds, so that the modal operators can be transformed into quantifiers
of classical logic. In particular, in the functional approach [2, 19], accessibility
is represented by means of functions, and the most common properties, such as
transitivity or reflexivity, can be taken into account by an equational unification
algorithm. An advantage of the functional method is that it keeps the structure
of the original formula. Hence, by adopting this method, modal Horn clauses may
be translated to first order formulas, and, by applying skolemization, to an Horn
clause language with equational unification.

In particular, an optimization of the functional translation method for the class
of inclusion logics has been proposed by Gasquet in [14]. Since we deal with a specific
language, namely with modal Horn clauses only containing universal modalities,
the case we consider can be regarded as a special instance of the one in [14]. In
particular, in the case when only ground terms may occur within modalities in
the program, in the goal and in the axioms (which is the one he considers), the
generality of equational unification may be replaced with a notion of matching (or
a notion of string rewriting). As a difference with [14] we deal with parametric
modalities. Hence, in the general case when variables may occur within modalities
we also need some form of equational unification, though for different reasons.

Also, the translation approach has been used in [8, 18] to obtain standard Pro-
log programs starting form Horn clauses extended with modal operators. In [8] the
functional translation method is extended to multimodal logic, and it is applied
to modal logic programming. The modalities considered are both universal and
existential, and are of any type among KD, KT, KD4, KT4, KF, and interaction
axioms of the form [a₁]α ⊃ [a₂]α are allowed. More general inclusion axioms, as
the ones we deal with are not considered. Nonnengart [18] has proposed a mixed
approach based on a relational and functional translation. One of his aims is to
avoid theory unification. As a particular case, modal Horn clauses may be directly
translated to Prolog clauses. This method requires that accessibility relation prop-
erties are first-order predicate logic definable. Moreover, if Prolog is to be used as
the first order inference machine, accessibility relation properties must be defined
through Horn clauses. In particular, he can provide Prolog translation for modal-
ities with the properties of KD, KT, KD4, S4, but he also can deal with axioms
like (B): α ⊃ ◻◇α, and hence with logics like KDB, KD45 and S5.

Acknowledgements

The authors would like to thank Floriano Zini. This work has been partially sup-
ported by MRST 40% “Intelligent Systems” project.

References


