Modal logic programming revisited

Linh Anh Nguyen

University of Warsaw
Institute of Informatics
ul. Banacha 2,
02-097 Warsaw (Poland)

nguyen@mimuw.edu.pl

ABSTRACT. *We present optimizations for the modal logic programming system MProlog, including the standard form for resolution cycles, optimized sets of rules used as meta-clauses, optimizations for the version of MProlog without existential modal operators, as well as iterative deepening search and tabulation. Our SLD-resolution calculi for MProlog in a number of modal logics are still strongly complete when resolution cycles are in the standard form and optimized sets of rules are used. We also show that the labelling technique used in our direct approach is relatively better than the Skolemization technique used in the translation approaches for modal logic programming.*

KEYWORDS: *modal logic, logic programming, MProlog.*

DOI:10.3166/JANCL.19.167–181 © 2009 Lavoisier, Paris

1. Introduction

Modal logic programming is the field that extends classical logic programming to deal with modalities. As modal logics can be used, among others, to reason about knowledge and belief, developing a good formalism for modal logic programs and an efficient computational procedure for it is desirable.

The first work on extending logic programming with modal logic is (Fariñas del Cerro, 1986) on the implemented system Molog (Fariñas del Cerro *et al.*, 1986). With Molog, the user can fix a modal logic and define or choose the rules to deal with modal operators. Molog can be viewed as a framework which can be instantiated with particular modal logics. As an extension of Molog, the Toulouse Inference Machine (Balbiani *et al.*, 1991) together with an abstract machine model called TARSKI for implementation (Balbiani *et al.*, 1992) makes it possible for a user to select clauses which cannot exactly unify with the current goal, but just resemble it in some way.

Apart from the mentioned works, modal logic programming has been studied by several authors in (Balbiani *et al.*, 1988; Akama, 1989; Debart *et al.*, 1992; Nonnengart, 1994; Baldoni *et al.*, 1996) and by us in (Nguyen, 2003; Nguyen, 2004; Nguyen, 2006). The works (Balbiani *et al.*, 1988; Baldoni *et al.*, 1996) and our mentioned works use the direct approach, the works (Akama, 1989; Debart *et al.*, 1992) use the functional translation approach, and the work (Nonnengart, 1994) uses the semi-functional approach. See (Orgun *et al.*, 1994; Nguyen, 2006) for further information.

In (Nguyen, 2006), we gave a framework for developing the least model semantics, fixpoint semantics, and SLD-resolution calculi for positive modal programs (called MProlog programs) in modal logics whose frame restrictions consist of the seriality conditions (i.e. $\forall x \exists y\, R_i(x, y)$ for every modal index i) and some classical Horn clauses. We have applied the framework for basic serial monomodal logics (Nguyen, 2003), multimodal logics of belief (Nguyen, 2006), the class $BSMM$ of basic serial multimodal logics (Nguyen, 2006), and the class $sCFG$ of serial context-free grammar logics (Nguyen, 2007).

The special feature of our framework is that it uses the direct approach and does not assume any special restriction on occurrences of modal operators[1], while the work (Balbiani *et al.*, 1988) assumes that universal modal operators do not occur in bodies of program clauses and goals, and the work (Baldoni *et al.*, 1996) assumes that existential modal operators do not occur in program clauses and goals.

Using our framework we have designed and implemented the modal logic programming system MProlog (Nguyen, 2004; Nguyen, 2008b). The theoretical foundation of the MProlog system is very different than that of Molog (Fariñas del Cerro *et al.*, 1986). In MProlog, the labelling technique is used for existential modal operators instead of Skolemization. Our system uses new technicalities like normal forms of modalities and pre-orders between modal operators. It also eliminates some drawbacks of Molog, e.g., MProlog gives substitutions as computed answers, while Molog can only answer “yes” or “no” (where “yes” means there exists a correct answer).

In this paper, we present new results and optimization techniques for modal logic programming. One of the theoretical results is the theorem about *strong* completeness of our SLD-resolution calculi for MProlog. We give various optimizations for MProlog that are both interesting from the theoretical point of view and useful for the implementation, including the standard form for resolution cycles, optimized sets of rules used as meta-clauses, optimizations for the version of MProlog without existential modal operators, as well as iterative deepening search and tabulation. The other theoretical results are that our SLD-resolution calculi for MProlog in a number of modal logics are still strongly complete when resolution cycles are in the standard form and optimized sets of rules are used. We also show that the labelling technique used in our direct approach is relatively better than the Skolemization technique used in the translation approaches for modal logic programming.

1. Programs and goals in our framework are of a normal form but the language is as expressive as the general modal Horn fragment.

This paper can be treated as a supplement to (Nguyen, 2006). We assume that the reader is familiar with modal logic and logic programming. Despite that we have made this paper self-contained to a certain extent, for a more comprehensive explanation we recommend the reader to read some parts of (Nguyen, 2006), e.g. the illustrating example given in (Nguyen, 2006, Section 1). Due to the lack of space, the proofs of the theorems given in this paper are presented only in the technical report (Nguyen, 2008a).

2. Preliminaries

2.1. *Considered modal logics*

Modal logics considered in our framework of modal logic programming are *quantified modal logics with fixed domain and rigid terms*. Their language is an extension of the language of classical first-order logic with modal operators $\Box_i$ and $\Diamond_i$, for $1 \leq i \leq m$ (where m is a fixed number). If $m = 1$ then we ignore the subscript i and write $\Box$ and $\Diamond$. The operators $\Box_i$ are called *universal* modal operators, while $\Diamond_i$ are called *existential* modal operators.

We restrict ourselves to modal logics that extend the quantified modal logic $K_{(m)}$ with axioms $(D) : \Box_i\varphi \to \Diamond_i\varphi$ (for $1 \leq i \leq m$) and some axioms that correspond to frame restrictions of the form of a Horn clause. The axiom (D) for modal index i corresponds to the seriality condition $\forall x \exists y\, R_i(x, y)$. When $\Box_i\varphi$ is read as "agent i believes that φ holds" (and $\Diamond_i\varphi$ is treated as $\neg\Box_i\neg\varphi$), the axiom says that beliefs of agent i are consistent.

In some examples we use the modal logics $KDI4_s$ and $KDI4_s5$. In these logics, $\Box_i\varphi$ means "φ is believed with degree at least i". These logics are axiomatised as:

$$\begin{aligned} KDI4_s &= K_{(m)} + (D) + (I) + (4_s) \\ KDI4_s5 &= K_{(m)} + (D) + (I) + (4_s) + (5) \end{aligned}$$

where the schemata of the additional axioms are: [2]

$$\begin{aligned} (I) :&\quad \Box_i\varphi \to \Box_j\varphi \ \text{ if } i > j, \\ (4_s) :&\quad \Box_i\varphi \to \Box_j\Box_i\varphi \quad \text{(strong positive introspection)}, \\ (5) :&\quad \neg\Box_i\varphi \to \Box_i\neg\Box_i\varphi \quad \text{(negative introspection)}. \end{aligned}$$

2.2. *The logical formalism MProlog*

A *universal modality* is a (possibly empty) sequence of universal modal operators. We use $\boxdot$ to denote a universal modality. Similarly as in classical logic programming,

2. Axiom $(5_s) : \neg\Box_i\varphi \to \Box_j\neg\Box_i\varphi$ (strong negative introspection) is derivable in $KDI4_s5$.

we write $\boxdot(\varphi \leftarrow \psi_1, \ldots, \psi_n)$ to denote the formula $\forall(\boxdot(\varphi \vee \neg\psi_1 \ldots \vee \neg\psi_n))$.[3] We use E to denote a classical atom.

A *program clause* is a formula of the form $\boxdot(A \leftarrow B_1, \ldots, B_n)$, where $n \geq 0$ and A, B_1, …, B_n are formulas of the form E, $\Box_i E$, or $\Diamond_i E$. $\boxdot$ is called the *modal context*, A the *head*, and $(B_1, \ldots, B_n)$ the *body* of the program clause. An *MProlog program* is a finite set of program clauses.

An *MProlog goal atom* is a formula of the form $\boxdot E$ or $\boxdot\Diamond_i E$, where $\boxdot$ is called the *modal context* of the goal atom. An *MProlog goal* is a formula written in the clausal form $\leftarrow \alpha_1, \ldots, \alpha_k$, where each α_i is an MProlog goal atom.

Let P be an MProlog program and G an MProlog goal of the form $\leftarrow \alpha_1, \ldots, \alpha_k$. A substitution θ is a *correct answer* in a modal logic *L* for $P \cup \{G\}$ if the domain of θ consists of variables occurring in G and $P \models_L \forall((\alpha_1 \wedge \ldots \wedge \alpha_k)\theta)$.

It is shown in (Nguyen, 2008a) that MProlog has the same expressive power as the general Horn fragment in normal modal logics that are characterised by a class of Kripke structures. For a specific modal logic *L*, we may adopt some restrictions on modal contexts of MProlog program clauses and MProlog goal atoms and call the obtained language *L-MProlog*. (If no restriction is adopted then *L*-MProlog is the same as MProlog.) Such restrictions either follow from equivalencies in *L* or are acceptable from the practical point of view, and furthermore, they do not reduce expressiveness of the language.

For example, in $KDI4_s5$ we have the equivalence $\nabla\nabla'\varphi \equiv \nabla'\varphi$, where ∇ and ∇' are arbitrary modal operators. Hence we can assume that the modal context of an $KDI4_s5$-*MProlog program clause* has length 1 or 0, and an $KDI4_s5$-*MProlog goal atom* is a formula of the form E, $\Box_i E$ or $\Diamond_i E$, with E being a classical atom.

2.3. *A framework of SLD-resolution for MProlog*

In (Nguyen, 2006), we gave a framework for developing fixpoint semantics, the least model semantics, and SLD-resolution calculi for *L*-MProlog, where *L* is a serial modal logic whose frame restrictions except seriality are Horn clauses (of classical first-order logic). We outline here the fragment involving SLD-resolution of that framework.

From now on, by a *modal operator* we mean $\Box_i$, $\Diamond_j$, or $\langle S\rangle_k$, where $\langle S\rangle_k$ is $\Diamond_k$ labelled by S, which is either a classical atom or a variable for classical atoms (called an *atom variable*). For further information on labelled modal operators, see (Nguyen, 2006). We use ∇ and ∇' to denote modal operators.

A *modality* is a (possibly empty) sequence of modal operators. We use $\triangle$ to denote a modality. Recall that we use E to denote a classical atom. A *modal atom* is a formula

3. By $\forall(\varphi)$ we denote the *universal closure* of φ, which is the formula obtained by adding a universal quantifier for every variable having a free occurrence in φ.

of the form $\triangle E$. A *simple modal atom* is a formula of the form E or ∇E. We use A, B to denote simple modal atoms, and α, β to denote modal atoms.

There may exist a compact form for modalities in *L*. For each specific modal logic *L*, we define *L-normal form of modalities*. For example, a modality is in $KDI4_s5$-normal form if its length is 0 or 1. It is possible that no restriction is adopted for *L*-normal form of modalities. A modality is in *L-normal labelled form* if it is in *L*-normal form and does not contain unlabelled existential modal operators $\Diamond_i$. A modal atom $\triangle E$ is in *L*-normal (labelled) form if $\triangle$ is in *L*-normal (labelled) form. Given a ground modal atom, the NF_L *operator* converts it to *L*-normal form.

Given a modal atom α, one can derive other modal atoms from α using axioms of *L*. The corresponding operator is called the Sat_L *operator* (where Sat stands for "saturation"). The direct consequence operator $T_{L,P}$ is defined using Sat_L and NF_L. An SLD-resolution calculus can be viewed as a reversed analogue of a direct consequence operator. Hence, to define an SLD-resolution calculus for *L*-MProlog we need reversed analogues of the operators Sat_L and NF_L. These operators are called the $rSat_L$ *operator* and the rNF_L *operator*, respectively. See (Nguyen, 2006) for formal definitions of the operators Sat_L, NF_L, $rSat_L$, and rNF_L.

The $rSat_L$ and rNF_L operators are each specified by a finite set of rules of the form $\alpha \leftarrow \beta$, where α and β are (schemata of) modal atoms. The rules are used as meta-clauses in SLD-derivations. Such rules can be accompanied by conditions which specify when the rule are applicable.[4]

For example, the rNF_L and $rSat_L$ operators for $L = KDI4_s5$ are specified by the following rules, where X is a fresh atom variable:[5]

$$\begin{array}{lll} rNF_L: & (a) & \nabla E \leftarrow \langle X\rangle_i \nabla E; \\ rSat_L: & (a) & \nabla\nabla' E \leftarrow \nabla' E, \\ & (b) & \Diamond_i E \leftarrow \Diamond_j E \text{ if } i > j, \\ & (c) & \Diamond_i E \leftarrow \langle X\rangle_i E. \end{array}$$

A *goal* is a clause of the form $\leftarrow \alpha_1, \ldots, \alpha_k$, where each α_i is a modal atom. *Resolvents* of a goal $\leftarrow \alpha_1, \ldots, \alpha_k$ and an $rSat_L/rNF_L$ rule $\alpha \leftarrow \beta$ are defined in the usual way (Nguyen, 2006). For example, resolving $\leftarrow \Box_1\Diamond_2 p(x)$ with the rule $\nabla\nabla' E \leftarrow \nabla' E$ results in $\leftarrow \Diamond_2 p(x)$, since ∇ is instantiated to $\Box_1$, and ∇' is instantiated to $\Diamond_2$.

For each specific modal logic *L*, we define a pre-order $\preceq_L$ to compare modal operators. For example, for $L = KDI4_s5$, the pre-order $\preceq_L$ is the least reflexive and transitive binary relation between modal operators such that: $\Diamond_i \preceq_L \langle S\rangle_i \preceq_L \Box_i$, and if $i < j$ then $\Box_i \preceq_L \Box_j$ and $\Diamond_j \preceq_L \Diamond_i$. If $\nabla \preceq_L \nabla'$ then we say that ∇ is an *L-instance* of ∇'. We say that an atom $\triangle E$ is an *L-instance* of $\triangle' E'$ if $\triangle$ and $\triangle'$ have

4. In general, a rule is of the form $(\alpha \leftarrow \varphi, \beta, \psi)$, where α and β stand for modal atoms, φ is a *pre-condition*, and ψ is a *post-computation*.

5. This means that *standardizing variables apart* is also needed for atom variables.

the same length k and there exists a substitution θ such that $E = E'\theta$ and the modal operator at position i of $\triangle$ is an *L*-instance of the modal operator at position i of $\triangle'\theta$ for every $1 \leq i \leq k$.

If $\boxdot$ and $\boxdot'$ are universal modalities, and furthermore, $\boxdot$ is the modal context of an *L*-MProlog program clause, then we say that $\boxdot'$ is an *L-context instance* of $\boxdot$ if $\boxdot\varphi \rightarrow \boxdot'\varphi$ is a theorem in *L* for an arbitrary φ. For example, $\Box_1$ is a $KDI4_s5$-context instance of $\Box_2$.

The *forward labelled form* of an atom α is the atom α' such that if α is of the form $\triangle\Diamond_i E$ then $\alpha' = \triangle\langle E\rangle_i E$, else $\alpha' = \alpha$. For example, the forward labelled form of $\Diamond_1 s(a)$ is $\langle s(a)\rangle_1 s(a)$.

Let $G = \leftarrow \alpha_1, \ldots, \alpha_i, \ldots, \alpha_k$ be a goal and $\varphi = \boxdot(A \leftarrow B_1, \ldots, B_n)$ a program clause. Then G' is *derived* from G and φ in *L* using a most general unifier (mgu) θ, and called an *L-resolvent* of G and φ, if the following conditions hold:

– $\alpha_i = \triangle' A'$, with $\triangle'$ in *L*-normal labelled form, is called the *selected atom.*
– $\triangle'$ is an *L*-instance of $\boxdot'$ which in turn is an *L*-context instance of $\boxdot$.
– θ is an mgu such that: $A'\theta$ has the same classical atom as $A\theta$, and $A'\theta$ is an *L*-instance of the forward labelled form of $A\theta$.
– G' is the goal $\leftarrow (\alpha_1, \ldots, \alpha_{i-1}, \triangle' B_1, \ldots, \triangle' B_n, \alpha_{i+1}, \ldots, \alpha_k)\theta$.

For example, the only $KDI4_s5$-resolvent of $\leftarrow \Box_1 p(x)$ and $\Box_2(p(x) \leftarrow \Diamond_2 q(x))$ is $\leftarrow \Box_1\Diamond_2 q(x)$ (here, $\boxdot = \Box_2$ and $\triangle' = \boxdot' = \Box_1$). As another example, the only $KDI4_s5$-resolvent of $\leftarrow \langle Y\rangle_1\langle X\rangle_1 r(x), \langle X\rangle_1 s(x)$ and $\Box_1(\Box_1 r(x) \leftarrow s(x))$ is $\leftarrow \langle Y\rangle_1 s(x), \langle X\rangle_1 s(x)$ (here, $\boxdot = \boxdot' = \Box_1$ and $\triangle' = \langle Y\rangle_1$).

SLD-derivation from an *L*-MProlog program and an *L*-MProlog goal in *L* is defined using two kinds of steps: a) deriving an *L*-resolvent of a goal and a variant of a program clause, b) deriving a resolvent of a goal and a variant of an $rSat_L/rNF_L$ rule. The notions of SLD-refutation and computed answer for *L*-MProlog are defined in the usual way. See Figure 1 for an example of SLD-refutation.

A *selection rule* is a function that maps an SLD-derivation with the last goal $\leftarrow \alpha_1, \ldots, \alpha_k$ to some atom α_i $(1 \leq i \leq k)$. Let P be an *L*-MProlog program, G an *L*-MProlog goal, and $\mathcal{R}$ a selection rule. An *SLD-refutation* of $P \cup \{G\}$ in *L via* $\mathcal{R}$ is an SLD-refutation of $P \cup \{G\}$ in *L* that uses $\mathcal{R}$ to select atoms.

Using the given framework, in (Nguyen, 2003; Nguyen, 2006; Nguyen, 2007) we have specified SLD-resolution calculi for *L*-MProlog in basic serial/almost serial monomodal logics, multimodal logics of belief, the class $BSMM$ of basic serial multimodal logics, and the class $sCFG$ of serial context-free grammar logics.

THEOREM 1. — *The SLD-resolution calculi given in (Nguyen, 2003; Nguyen, 2006; Nguyen, 2007) for L-MProlog in various modal logics L are sound and strongly complete:*

Soundness: *Let P be an L-MProlog program and G an L-MProlog goal. Then every computed answer in L for $P \cup \{G\}$ is a correct answer in L for $P \cup \{G\}$.*

Consider the goal $G = \leftarrow \Box_1 p(x)$ and the following program P:

$$\begin{array}{lcl}
\varphi_1 & = & \Box_2(p(x) \leftarrow \Diamond_2 q(x)) \\
\varphi_2 & = & \Box_1(q(x) \leftarrow r(x), s(x)) \\
\varphi_3 & = & \Box_1(\Box_1 r(x) \leftarrow s(x)) \\
\varphi_4 & = & \Diamond_1 s(a) \leftarrow
\end{array}$$

Here is an SLD-refutation of $P \cup \{G\}$ in $L = KDI4_s5$:

Goals	Input clauses/rules	MGUs
$\leftarrow \Box_1 p(x)$		
$\leftarrow \Box_1 \Diamond_2 q(x)$	φ_1	$\{x_1/x\}$
$\leftarrow \Diamond_2 q(x)$	$rSat_L(a)$	
$\leftarrow \Diamond_1 q(x)$	$rSat_L(b)$	
$\leftarrow \langle X \rangle_1 q(x)$	$rSat_L(c)$	
$\leftarrow \langle X \rangle_1 r(x), \langle X \rangle_1 s(x)$	φ_2	$\{x_5/x\}$
$\leftarrow \langle Y \rangle_1 \langle X \rangle_1 r(x), \langle X \rangle_1 s(x)$	$rNF_L(a)$	
$\leftarrow \langle Y \rangle_1 s(x), \langle X \rangle_1 s(x)$	φ_3	$\{x_7/x\}$
$\leftarrow \langle X \rangle_1 s(a)$	φ_4	$\{x/a, Y/s(a)\}$
the empty clause	φ_4	$\{X/s(a)\}$

(Variables of the input clause of step i are subscripted by i.)

Figure 1. *An example of SLD-resolution for MProlog*

Strong completeness: *Let P be an L-MProlog program, G an L-MProlog goal, and $\mathcal{R}$ a selection rule. For every correct answer θ in L of $P \cup \{G\}$, there exists an SLD-refutation of $P \cup \{G\}$ in L via $\mathcal{R}$ with computed answer γ such that $G\theta = G\gamma\delta$ for some substitution δ.*

See (Nguyen, 2003; Nguyen, 2006; Nguyen, 2008a) for the proofs of this theorem.

3. Optimizations

3.1. *The standard form for resolution cycles*

Roughly speaking, an SLD-derivation from a program P and a goal G in L is an application of a sequence of program clauses of P and $rSat_L/rNF_L$ rules to the goal G. (By an application of a program clause or a rule we mean an application of its variant.) A *resolution cycle* is defined to be a fragment of an SLD-derivation that starts either immediately after an application of a program clause or from the beginning of the derivation, and ends with an application of a program clause. Note that an SLD-refutation can be divided into a sequence of resolution cycles.

A *selection rule is standard* if in every resolution cycle only atoms at the same position are selected. A resolution cycle in an SLD-derivation via a standard selection function is thus an application of a sequence of $rSat_L/rNF_L$ rules with a program clause at the end to the selected atom.

A resolution cycle is in the *standard form* if it is a resolution cycle via a standard selection function with the property that, $rSat_L$ rules are applied before rNF_L rules, which in turn are applied before the used program clause.[6]

THEOREM 2. — *The SLD-resolution calculi given in (Nguyen, 2003; Nguyen, 2006; Nguyen, 2007) for L-MProlog in various modal logics L are still strongly complete when adopting the restriction that resolution cycles are in the standard form.*

See (Nguyen, 2008a, Section 5.1) for the proof of this theorem.

3.2. *Optimizing the set of* $rSat_L$ *rules*

Consider, for example, the SLD-resolution calculus given in (Nguyen, 2006) for $KDI4_s$-MProlog. This calculus uses the following $rSat_{KDI4_s}$ rules:

$$\begin{array}{l}
\triangle\Diamond_i E \leftarrow \triangle\langle X\rangle_i E \\
\triangle\nabla_i \alpha \leftarrow \triangle\Box_j \alpha \ \text{ if } i \le j \\
\triangle\Diamond_i E \leftarrow \triangle\Diamond_j E \ \text{ if } i > j \\
\triangle\nabla\Box_i \alpha \leftarrow \triangle\Box_i \alpha \\
\triangle\Diamond_i E \leftarrow \triangle\langle X\rangle_j \Diamond_i E
\end{array}$$

Recall that E stands for a classical atom, while α stands for a modal atom (of the form $\triangle E$). As shown in (Nguyen, 2008a), without loss of (strong) completeness of the SLD-resolution calculus for $KDI4_s$-MProlog, the occurrences of α in the above rules can be replaced by E (a more restricted form). The change results in a more efficient SLD-resolution calculus for $KDI4_s$-MProlog.

The intuition behind such an optimization is based on properties of the Sat_L operator of the fixpoint semantics. When constructing a least *L*-model for P and realizing a formula of the form $(A \leftarrow B_1, \ldots, B_n)$ at a possible world w, to check $B_1, \ldots, B_n$ at w, or equivalently, to derive modal atoms $\triangle B_1, \ldots, \triangle B_n$, where $\triangle = w$, only Sat_L rules that manipulate *suffixes* of modal atoms are needed. This kind of optimization for the set of rules specifying Sat_L makes a corresponding optimization for the set of rules specifying $rSat_L$.

The mentioned optimization is applicable for SLD-resolution calculi of *L*-MProlog in a number of other modal logics. See (Nguyen, 2008a, Section 5.2) for a proof of the following theorem.

6. Note that a resolution cycle may use no $rSat_L/rNF_L$ rules. The operator rNF_L is usually deterministic and one application of an rNF_L rule is usually enough for one resolution cycle.

THEOREM 3. — *Let L be any serial monomodal logic considered in (Nguyen, 2003) or any multimodal logic of belief considered in (Nguyen, 2006). By changing every $rSat_L$ rule of the form $\triangle\triangle'\alpha \leftarrow \triangle\triangle''\alpha$ of the SLD-resolution calculus given in (Nguyen, 2003; Nguyen, 2006) for L-MProlog to the more restricted form $\triangle\triangle' E \leftarrow \triangle\triangle'' E$, the obtained SLD-calculus is still sound and strong complete for L-MProlog (independently from whether resolution cycles are required to be in the standard form or not).*

The mentioned optimization, however, is not applicable for the SLD-resolution calculi given in (Nguyen, 2006; Nguyen, 2007) for MProlog in the large classes $BSMM$ and $sCFG$ of multimodal logics. The reason is that, a chain of applications of $rSat_L$ rules for $L \in \{BSMM, sCFG\}$ may be too complex, and we may need to apply the first rules of that chain on the middle of the considered modality.

As another optimization, we can avoid explicit use of the (general) $\Box$*-lifting rule* "$\triangle\nabla\alpha \leftarrow \triangle\Box_i\alpha$ if $\nabla \preceq_L \Box_i$" by embedding it into the other $rSat_L$ rules. For example, the optimized set of $rSat_{KDI4_s}$ rules consists of the following:

$$\begin{array}{l}
\triangle\Diamond_i E \leftarrow \triangle\langle X\rangle_i E \\
\triangle\Diamond_i E \leftarrow \triangle\Diamond_j E \ \text{ if } i > j \\
\triangle\nabla\nabla'_i E \leftarrow \triangle\Box_i E \\
\triangle\Diamond_i E \leftarrow \triangle\langle X\rangle_j\Diamond_i E
\end{array}$$

where ∇'_i denotes an arbitrary modal operator with index i. See (Nguyen, 2008a) for optimized sets of $rSat_L$ rules for *L*-MProlog in other modal logics.

3.3. *The case of MProlog-$\Box$*

An *L-MProlog-$\Box$ program* (resp. *goal*) is an *L*-MProlog program (resp. goal) without existential modal operators. In this subsection, let L denote one of the modal logics considered in (Nguyen, 2003; Nguyen, 2006; Nguyen, 2007) except $BSMM$. Consider the original or optimized SLD-resolution calculus given in (Nguyen, 2003; Nguyen, 2006; Nguyen, 2007; Nguyen, 2008a) for *L*-MProlog. We show that the calculus can be significantly simplified for *L*-MProlog-$\Box$.

Let P be an *L*-MProlog-$\Box$ program and G an *L*-MProlog-$\Box$ goal. Observe that:

1) For every rule specifying $rSat_L$ or rNF_L, if the r.h.s. atom contains an unlabelled existential modal operator then the l.h.s. atom also contains an unlabelled existential modal operator. Hence all the $rSat_L$ rules containing unlabelled existential modal operators can be deleted without affecting SLD-refutations of $P \cup \{G\}$ in *L*.

2) If we replace the operators $\langle X\rangle_i$, $\langle X\rangle_j$, and $\langle X\rangle$ in the rules specifying rNF_L respectively by $\Box_i$, $\Box_j$, and $\Box$, then all SLD-refutations of $P \cup \{G\}$ change accordingly and remain correct. So, we can assume this replacement for *L*-MProlog-$\Box$. Similarly, we can replace the $rSat_L$ rule $\triangle\alpha \leftarrow \triangle\langle X\rangle\Box\alpha$ for $L \in \{KDB, B\}$ by $\triangle\alpha \leftarrow \triangle\Box\Box\alpha$.

3) With the modifications mentioned in the two preceding items, every SLD-derivation from $P \cup \{G\}$ in L does not contain any (labelled or unlabelled) existential modal operator. Consequently, modal operators of the form ∇_i in the (remaining) rules specifying $rSat_L$ or rNF_L can be replaced by $\Box_i$. In general, the rules can be reformulated so that they contain no operators of the form ∇.

The mentioned modifications create a more efficient, sound and strong complete SLD-resolution calculus for L-MProlog-$\Box$. For example, we need only one $rSat_{KDI4_s}$ rule $\triangle\Box_j\Box_i E \leftarrow \triangle\Box_i E$ (and no rNF_{KDI4_s} rules) for $KDI4_s$-MProlog-$\Box$. As another example, for $KDI4_s5$-MProlog-$\Box$ we need only the following rules:

$$\begin{array}{l} rNF_{KDI4_s5} : \Box_i E \leftarrow \Box_j\Box_i E \\ rSat_{KDI4_s5} : \Box_j\Box_i E \leftarrow \Box_i E \end{array}$$

3.4. *Restrictions and iterative deepening search*

In this and the next subsection, we present additional optimizations that have been implemented for the modal logic programming MProlog version 2.0 (Nguyen, 2008b).

It is not easy for users to imagine and control the behaviour of MProlog programs. One of the reasons is that MProlog uses $rSat_L/rNF_L$ rules as meta clauses and the users may not be fully aware of all possible effects of the rules. Even when they understand the rules well, they may not have enough control on the rules without modifying the interpreter of the used MProlog system. Another reason is that a complete SLD-resolution for L-MProlog may need a rule like $\triangle\Box_i E \leftarrow \triangle\Box_i\Box_i E$, which can be applied repeatedly forever. This difficulty suggests that programming in MProlog is better treated as a mixture of "programming" and theorem proving for the modal Horn fragment.

To restrict the search space for MProlog, one can apply some restrictions like:

1) a limit on the lengths of modalities that can occur in derivations,
2) a limit on the number of applications of $rSat_L$ (resp. rNF_L) rules in a resolution cycle,
3) a limit on the nesting depth of function symbols occurring in modal atoms,
4) a limit on the length of derivation,
5) a limit on the number of applications of $rSat_L/rNF_L$ rules in a derivation.

These restrictions may affect completeness of the used calculus. For real applications, however, it is reasonable to set the limits mentioned in the first three items of the above list to some low values. The remaining limits can be dealt with by iterative deepening search. For example, iterative deepening search w.r.t. the number of applications of $rSat_L/rNF_L$ rules in a derivation can be done by ignoring the limit on the length of derivation (or setting it to a high enough value), and at each deepening iteration, increasing the limit on the number of applications of $rSat_L/rNF_L$ rules by a certain value (e.g. by a constant specified by a parameter).

3.5. *Tabulation*

Setting the limits mentioned in the previous subsection to some concrete values, using the depth first search strategy an execution of an MProlog program may still loop forever as in the case of Prolog. A solution for this is to use some tabulation (tabling) mechanism.

There are advanced tabulation methods for Prolog like OLDT-resolution (Tamaki *et al.*, 1986), linear tabulated resolution (Shen *et al.*, 2001; Zhou *et al.*, 2003). These methods use sophisticated techniques (e.g. the suspension-resumption mechanism and the stack-wise representation of OLDT) that are better implemented by the underlying Prolog abstract machine. Besides, these methods try to reach "answer completion" for subgoals as early as possible.

For the MProlog system (version 2.0), which has been written in Prolog as a module, we adopted a tabulation mechanism with the essential feature that, looping for answer completion is done only at the most outer level for the top goal. The main loop continues when more answers have been tabulated (for some subgoals) during the last iteration. At each iteration of the loop, when the system encounters a subgoal that has a variant called earlier during the iteration, the subgoal is resolved only with the tabulated answers of its variant. As applications of different sequences of $rSat_L/rNF_L$ rules may give the same effect, our tabulation method that delays answer completion for subgoals has an advantage in quickly finding the first answer for the top goal.

EXAMPLE 4. — Consider the goal $G = \leftarrow \Diamond s(x)$ and the following program P in the modal logic *KD*:

$$
\begin{array}{lcl}
\varphi_1 & = & \Diamond p(x) \leftarrow q(x) \\
\varphi_2 & = & \Diamond p(x) \leftarrow r(x) \\
\varphi_3 & = & q(a) \leftarrow \\
\varphi_4 & = & r(a) \leftarrow \\
\varphi_5 & = & \Box(s(x) \leftarrow p(x), t(x), u(x)) \\
\varphi_6 & = & \Box(t(x) \leftarrow p(x))
\end{array}
$$

Here is an SLD-derivation from $P \cup \{G\}$ in *KD*:

Goals	Input clauses/rules
$G \ = \leftarrow \Diamond s(x)$	
$G_1 = \leftarrow \langle Y \rangle s(x)$	$rSat_{KD}$
$G_2 = \leftarrow \langle Y \rangle p(x), \langle Y \rangle t(x), \langle Y \rangle u(x)$	φ_5
$G_3 = \leftarrow q(x), \langle p(x) \rangle t(x), \langle p(x) \rangle u(x)$	φ_1
$G_4 = \leftarrow \langle p(a) \rangle t(a), \langle p(a) \rangle u(a)$	φ_3
$G_5 = \leftarrow \langle p(a) \rangle p(a), \langle p(a) \rangle u(a)$	φ_6
$G_6 = \leftarrow q(a), \langle p(a) \rangle u(a)$	φ_1
$G_7 = \leftarrow \langle p(a) \rangle u(a)$	φ_3
failure	

At the failure with G_7 the system backtracks to G_2 and resolves the goal atom $\langle Y\rangle p(x)$ with the next program clause φ_2, resulting in:

Goals	Input clauses/rules
$G_3^\star = \leftarrow r(x), \langle p(x)\rangle t(x), \langle p(x)\rangle u(x)$	φ_2
$G_4^\star = \leftarrow \langle p(a)\rangle t(a), \langle p(a)\rangle u(a)$	φ_4

As the goal atom $\langle p(a)\rangle t(a)$ has been called earlier and succeeded (the information remains through the mentioned backtracking), the next goal is $G_5^\star = \leftarrow \langle p(a)\rangle u(a)$, which fails. At this point, the systems backtracks to the top goal G. Since some new answers for subgoals (namely, $q(a)$, $\langle p(a)\rangle p(a)$, $\langle p(a)\rangle t(a)$) have been tabulated, the system makes another round for solving G after deleting information about what subgoals have been called. This round does not tabulate any new answer for subgoals, so when the system backtracks again to the top goal G, it stops with failure result. $\Box$

4. On the usefulness of the direct approach

The relationship between the direct approach and the translation approaches (Debart *et al.*, 1992; Nonnengart, 1994) for modal logic programming has been discussed in our papers (Nguyen, 2003; Nguyen, 2006). Further information can be found in (Nguyen, 2008a). In this section, we give some additional remarks on the usefulness of the direct approach.

As shown in our works (Nguyen, 2003; Nguyen, 2006), the direct approach manages very well with modal logics with variants of axiom (5). As discussed in (Ohlbach, 1988), functional translation cannot deals with axiom (5) in a pure way. Hence it is not straightforward to extend the functional translation approach by Debart *et al.* for programming in modal logics with axiom (5). The semi-functional translation approach by Nonnengart does not share this problem, but note that it produces atoms of the accessibility relations (for translating universal modal operators), which are loosely related with atoms of the other predicates. Hence, it is harder to use the semi-functional translation approach to develop an efficient search procedure for modal logic programming.

In the rest of this section, we argue that the labelling technique used in our direct approach is relatively better than the Skolemization technique used in the translation approaches for modal logic programming.

Consider the following logic program P in the modal logic $L = \mathit{KD}$:

$$\begin{array}{ll} \Diamond p \leftarrow q & \quad q \leftarrow \\ \Diamond p \leftarrow r & \quad r \leftarrow \end{array}$$

The direct consequence operator $T_{L,P}$ of our fixpoint semantics has the least fixpoint $\{q, r, \langle p\rangle p\}$. The functional translation of P results in the following program P':

$$\begin{array}{ll} p(\varepsilon!a) \leftarrow q(\varepsilon) & \quad q(\varepsilon) \leftarrow \\ p(\varepsilon!b) \leftarrow r(\varepsilon) & \quad r(\varepsilon) \leftarrow \end{array}$$

The semi-functional translation of P results in $P'' = P' \cup \{R(x, x!y) \leftarrow\}$.[7] The direct consequence operator of P' has the least fixpoint $\{q(\varepsilon),\ r(\varepsilon),\ p(\varepsilon!a),\ p(\varepsilon!b)\}$. Thus, the bottom-up computation for P' or P'' gives two atoms $p(\varepsilon!a)$ and $p(\varepsilon!b)$ instead of one atom $\langle p\rangle p$ that is created by our bottom-up computation for P. As shown below, this problem corresponds to another problem in top-down computation.

Reconsider the program P and the goal G given in Example 4. Functional translation of G and P gives $G' = \leftarrow s(\varepsilon!z, x)$ and the following program P':

$$\begin{aligned}
\psi_1 &= p(\varepsilon!f(x), x) \leftarrow q(\varepsilon, x) \\
\psi_2 &= p(\varepsilon!g(x), x) \leftarrow r(\varepsilon, x) \\
\psi_3 &= q(\varepsilon, a) \leftarrow \\
\psi_4 &= r(\varepsilon, a) \leftarrow \\
\psi_5 &= s(\varepsilon!y, x) \leftarrow p(\varepsilon!y, x), t(\varepsilon!y, x), u(\varepsilon!y, x) \\
\psi_6 &= t(\varepsilon!y, x) \leftarrow p(\varepsilon!y, x)
\end{aligned}$$

Consider SLD-resolution with tabulation for $P' \cup \{G'\}$. The first derivation is:

Goals	Input clauses
$G' = \leftarrow s(\varepsilon!z, x)$	
$G'_1 = \leftarrow p(\varepsilon!z, x), t(\varepsilon!z, x), u(\varepsilon!z, x)$	ψ_5
$G'_2 = \leftarrow q(\varepsilon, x), t(\varepsilon!f(x), x), u(\varepsilon!f(x), x)$	ψ_1
$G'_3 = \leftarrow t(\varepsilon!f(a), a), u(\varepsilon!f(a), a)$	ψ_3
$G'_4 = \leftarrow p(\varepsilon!f(a), a), u(\varepsilon!f(a), a)$	ψ_6
$G'_5 = \leftarrow q(\varepsilon, a), u(\varepsilon!f(a), a)$	ψ_1
$G'_6 = \leftarrow u(\varepsilon!f(a), a)$	ψ_3
failure	

At the failure with G'_6 the system backtracks to G'_1 and resolves the goal atom $p(\varepsilon!z, x)$ with the next program clause ψ_2, resulting in:

Goals	Input clauses
$G''_2 = \leftarrow r(\varepsilon, x), t(\varepsilon!g(x), x), u(\varepsilon!g(x), x)$	ψ_1
$G''_3 = \leftarrow t(\varepsilon!g(a), a), u(\varepsilon!g(a), a)$	ψ_3
$G''_4 = \leftarrow p(\varepsilon!g(a), a), u(\varepsilon!g(a), a)$	ψ_6
$G''_5 = \leftarrow q(\varepsilon, a), u(\varepsilon!g(a), a)$	ψ_1
$G''_6 = \leftarrow u(\varepsilon!g(a), a)$	ψ_3
failure	

The problem is that tabulation does not work as it did in Example 4 for the direct approach, because $t(\varepsilon!g(a), a)$ is not the same as $t(\varepsilon!f(a), a)$. That is, using the functional translation approach, more computation is needed for this example.

7. (Debart *et al.*, 1992) uses '!' to construct *path expressions*, while (Nonnengart, 1994) uses ':'.

The same problem occurs for the semi-functional translation approach. We give below the translation for the considered program P and goal G and leave detailed analysis for the reader. The results of the translation (using the modal logic *KD*) are the goal $G'' = \leftarrow s(z,x), R(\varepsilon,z)$ and the following program P'':

$$\begin{array}{l}
p(\varepsilon!f(x),x) \leftarrow q(\varepsilon,x) \\
p(\varepsilon!g(x),x) \leftarrow r(\varepsilon,x) \\
q(\varepsilon,a) \leftarrow \\
r(\varepsilon,a) \leftarrow \\
s(y,x) \leftarrow p(y,x), t(y,x), u(y,x), R(\varepsilon,y) \\
t(y,x) \leftarrow p(y,x), R(\varepsilon,y) \\
R(x,x!y) \leftarrow
\end{array}$$

5. Concluding remarks

We have presented significant optimizations for the theory of MProlog. They are very useful for increasing efficiency of the implemented system MProlog. We have also shown that the labelling technique used in our direct approach is relatively better than the Skolemization technique used in the translation approaches.

There are some problems deserving investigation: i) developing semantics for modal logic programs with negation, ii) developing a good methodology for practical programming in modal logics (i.e. a methodology with conceivable behaviour of modal logic programs), iii) applying modal logic programming for practical problems.

Acknowledgements

This work has been supported by grant N N206 3982 33 from the Polish Ministry of Science and Higher Education. I would like to thank also Professor Andrzej Szałas for reading the draft of this paper and giving useful comments.

6. References

Akama S., "A Meta-Logical Foundation of Modal Logic Programming", 1-20-1, Higashi-Yurigaoka, Asao-ku, Kawasaki-shi, 215, Japan, 1989.

Balbiani P., Fariñas del Cerro L., Herzig A., "Declarative Semantics for Modal Logic Programs", *Proceedings of the 1988 International Conference on Fifth Generation Computer Systems*, ICOT, pp. 507–514, 1988.

Balbiani P., Herzig A., Lima-Marques M., "TIM: The Toulouse Inference Machine for Non-Classical Logic Programming", *PDK'91: International Workshop on Processing Declarative Knowledge*, Springer-Verlag, pp. 365–382, 1991.

Balbiani P., Herzig A., Lima-Marques M., “Implementing Prolog Extensions: A Parallel Inference Machine”, *Proceedings of the 1992 International Conference on Fifth Generation Computer Systems*, ICOT, pp. 833–842, 1992.

Baldoni M., Giordano L., Martelli A., “A Framework for a Modal Logic Programming”, *Joint International Conference and Symposium on Logic Programming*, MIT Press, pp. 52–66, 1996.

Debart F., Enjalbert P., Lescot M., “Multimodal Logic Programming Using Equational and Order-Sorted Logic”, *Theoretical Computer Science*, vol. 105, pp. 141–166, 1992.

Fariñas del Cerro L., “MOLOG: A System that Extends PROLOG with Modal Logic”, *New Generation Computing*, vol. 4, pp. 35–50, 1986.

Fariñas del Cerro L., Herzig A., “MOLOG - a tool for non-classical logic programming”, `http://www.irit.fr/ACTIVITES/EQ_ALG/Herzig/molog.html`, 1986.

Nguyen L. A., “A Fixpoint Semantics and an SLD-Resolution Calculus for Modal Logic Programs”, *Fundamenta Informaticae*, vol. 55, num. 1, pp. 63–100, 2003.

Nguyen L. A., “The Modal Logic Programming System MProlog”, *in* J. Alferes, J. Leite (eds), *Proceedings of JELIA 2004, LNCS 3229*, Springer, pp. 266–278, 2004.

Nguyen L. A., “Multimodal Logic Programming”, *Theoretical Computer Science*, vol. 360, pp. 247–288, 2006.

Nguyen L. A., “Foundations of Modal Deductive Databases”, *Fundamenta Informaticae*, vol. 79, num. 1–2, pp. 85–135, 2007.

Nguyen L. A., “Foundations of Modal Logic Programming: The Direct Approach (release 2.0)”, manuscript (provided as a technical report), available at `http://www.mimuw.edu.pl/~nguyen/papers.html`, 2008a.

Nguyen L. A., “Source Files, Calculi, and Examples of MProlog (version 2.0)”, available at `http://www.mimuw.edu.pl/~nguyen/mprolog`, 2008b.

Nonnengart A., “How to Use Modalities and Sorts in Prolog”, *in* C. MacNish, D. Pearce, L. Pereira (eds), *Proceedings of JELIA’94, LNCS 838*, Springer, pp. 365–378, 1994.

Ohlbach H., “A Resolution Calculus for Modal Logics”, *Proceedings of CADE-88, LNCS 310*, Springer, pp. 500–516, 1988.

Orgun M. A., Ma W., “An Overview of Temporal and Modal Logic Programming”, *D.M. Gabbay and H.J. Ohlbach, editors, Proc. First Int. Conf. on Temporal Logic - LNAI 827*, Springer-Verlag, pp. 445–479, 1994.

Shen Y.-D., Yuan L.-Y., You J.-H., Zhou N.-F., “Linear Tabulated Resolution Based on Prolog Control Strategy”, *TPLP*, vol. 1, num. 1, pp. 71–103, 2001.

Tamaki H., Sato T., “OLD Resolution with Tabulation”, *in* E. Shapiro (ed.), *Proceedings of ICLP’1986, LNCS 225*, Springer, pp. 84–98, 1986.

Zhou N.-F., Sato T., “Efficient Fixpoint Computation in Linear Tabling”, *Proceedings of PPDP’2003*, ACM, pp. 275–283, 2003.