Foundations of Modal Deductive Databases*

Linh Anh Nguyen[†]

Institute of Informatics, University of Warsaw
ul. Banacha 2, 02-097 Warsaw, Poland
nguyen@mimuw.edu.pl

November 2006 (last revised: May 2010)

Abstract

We give formulations for modal deductive databases and present a modal query language called MDatalog. We define modal relational algebras and give the seminaive evaluation algorithm, the top-down evaluation algorithm, and the magic-set transformation for MDatalog queries. The results of this paper like soundness and completeness of the top-down evaluation algorithm or correctness of the magic-set transformation are proved for the multimodal logics of belief $KDI4_s5$, $KDI45$, $KD4_s5_s$, $KD45_{(m)}$, $KD4I_g5_a$, and the class of serial context-free grammar logics. We also show that MDatalog has PTIME data complexity in the logics $KDI4_s5$, $KDI45$, $KD4_s5_s$, and $KD45_{(m)}$.

1 Introduction

Modal logics are useful for reasoning about knowledge and beliefs. In particular, they can be used for reasoning about multi-degree beliefs (a kind of uncertainty), for modeling distributed systems of beliefs, and for reasoning about epistemic states of agents in multi-agent systems (see Section 2.2 for some modal logics with these purposes). Example 2.1 in this paper demonstrates the usefulness of modal logics in modeling distributed systems of beliefs. Example 4.3 in [25] about the wise men puzzle demonstrates how agents can use modal logics to reason about common beliefs and epistemic states of other agents.

Deductive databases are very useful for practical applications. In deductive databases, intentional relations are defined using extensional relations and logical rules, and users can thus create sophisticated relations from basic ones. The field of deductive databases is mature and there are well-developed techniques for computing queries in such databases (see, e.g., [1]). It is desirable to study modal extensions of deductive databases.

As far as we know, there are no works by other authors that are devoted to modal deductive databases and directly use modal logics. The modal Datalog defined in [13, Definition 23] is formulated in classical logic and uses only unary or binary predicates. Deductive databases/knowledge bases in description logics have recently been studied by a considerable number of researchers (see, e.g., [4, 17, 14, 10, 15, 5]). Description logics are cousins of modal logics that represent the domain of interest in terms of concepts, objects, roles, and are useful for modeling and reasoning about structured knowledge. However, they differ from modal logics in the same way as "objects" differ from "possible worlds".

Not only are modal extensions desirable for deductive databases, but the computational methods of deductive databases are also worth studying for modal logics. Informally, assume that a modal query consists of a first-order positive modal logic program without function symbols and a querying

*This is a revised version of "L.A. Nguyen. *Foundations of Modal Deductive Databases.* Fundamenta Informaticae, 79 (1-2), 2007, 85-135."

[†]Supported by grant N N206 3982 33 from the Polish Ministry of Science and Higher Education.

formula of the form $\exists \overline{x}\, \varphi$, where φ is a positive formula without quantifiers and $\overline{x}$ is a vector of all the variables of φ. If one applies the relational translation to classical logic for a modal query, then Skolem function symbols may be introduced and the Horn property may not be preserved. If the functional translation [29, 7] or the semi-functional translation [28] is used, then Skolem function symbols may appear. That is, translation techniques do not work for modal queries. Also note that the method of systematically checking all ground substitutions is very insufficient and unacceptable. Therefore the study of the computational methods of deductive databases for modal queries is fully justifiable.

In [20], we extended Datalog for monomodal logics, giving two languages: MDatalog and eMDatalog. The first one is a natural extension of Datalog, while the second one is the general modal Horn fragment with a refined condition of range-restrictedness. It was shown in [20] that every eMDatalog query can be translated in polynomial time into an MDatalog query which is equivalent to the original in any normal monomodal logic. The evaluation method proposed in [20] is based on building a least L-model for a modal deductive database, where L is the base modal logic. It was applied for the basic serial/almost serial monomodal logics. In [24], we showed that the data complexity of MDatalog and eMDatalog in KD, T, KB, KDB, B, $K5$, $KD5$, $K45$, $KD45$, $KB5$, and $S5$ is complete in PTIME, in K is complete in coNP, and in $K4$, $KD4$, and $S4$ is complete in PSPACE. This implies that, in general, MDatalog and eMDatalog are more expressive than Datalog, since the data complexity of Datalog is complete in PTIME. Recall also that if one translates an MDatalog program into a logic program in classical first-order logic then Skolem function symbols may need to be used and the resulting program is not a Datalog program.

In [21, 25, 27], we proposed a modal logic programming language called MProlog and gave a framework for developing least model semantics, fixpoint semantics, and SLD-resolution calculi for MProlog programs. Our framework uses a direct approach and does not assume any special restriction on the form of program clauses and goals. The framework has been applied for basic serial monomodal logics, multimodal logics of belief, a class of basic serial multimodal logics, and the class of serial context-free grammar logics. A prototyping implementation of MProlog was reported in [22]. In the works [21, 25, 27] on MProlog, we used a special structure called a model generator to represent a Kripke model. A model generator is a set of ground modal atoms, which may contain labeled existential modal operators. The direct consequence operator of the fixpoint semantics is a function that maps a model generator to another one. With that feature, we are able to group atoms in a model generator by predicate symbols and this is a key to develop modal relational algebras. Furthermore, as we will see, the fixpoint semantics and SLD-resolution calculi for MProlog are starting points for developing evaluation methods for modal deductive databases.

In our recent conference paper [23], we extended the query language MDatalog for multimodal deductive databases. Basing on the existing techniques of Datalog, we defined modal relational algebras and gave the seminaive evaluation algorithm and the magic-set transformation for MDatalog queries. These bottom-up evaluation techniques for MDatalog closely relate to the fixpoint semantics of MProlog programs. The results of [23] were presented for the multimodal logics $KDI4_s5$, $KDI45$, $KD4_s5_s$, $KD45_{(m)}$, which are multimodal extensions of the monomodal logic $KD45$. The logics $KDI4_s5$ and $KDI45$ are intended for reasoning about multi-degree belief, while $KD4_s5_s$ can be used for distributed systems of belief, and $KD45_{(m)}$ can be used for reasoning about epistemic states of agents. We proved that MDatalog has PTIME data complexity in these logics.

This work is a revised and extended version of the above mentioned conference paper. In this work, we present our results also for the multimodal logic $KD4I_g5_a$ and the class $sCFG$ of serial context-free grammar logics. The logic $KD4I_g5_a$ is intended for reasoning about belief and common belief of agents [26]. The class $sCFG$ contains serial multimodal logics with axioms of the form $\Box_i\varphi \to \Box_{j_1} \ldots \Box_{j_h}\varphi$. We give a full proof of correctness of the magic-set transformation for MDatalog in the considered modal logics (which was not included in [23]). Furthermore, we present a top-down evaluation algorithm for MDatalog, which is based on our SLD-resolution calculi for MProlog, and prove its soundness, completeness, and tightness. Our algorithm relates to the query-subquery evaluation method of Datalog, but we do not use adornments and annotations. Our top-down evaluation method is more efficient than the bottom-up evaluation method based on our magic-set transformation and the seminaive evaluation. (Our magic-set transformation for MDatalog does not strictly simulate our top-down evaluation algorithm because modal contexts of goal atoms cannot be pushed from goals to subgoals in a pure way and for that reason we allow some loosening.)

The rest of this work is structured as follows. In Section 2, we give basic definitions for multimodal logics, introduce multimodal logics of belief and the class *sCFG* of serial context-free grammar logics, give an ordering between Kripke models, and define the modal logic programming language MProlog. In Section 3, we recall our framework of [25] for developing fixpoint semantics and SLD-resolution calculi for MProlog programs. We also present our instantiations of the framework for the considered modal logics. Proofs of correctness of the instantiations are not included in this work but they can be found in [25, 27]. In Section 4, we define the MDatalog language and give definitions for modal deductive databases. In Section 5, we show that MDatalog has PTIME data complexity in $KDI4_s5$, $KDI45$, $KD4_s5_s$, $KD45_{(m)}$. In Section 6, we define the L-SPCU algebra, which is an extension of the relational algebra SPCU for a modal logic L, and show that nonrecursive MDatalog queries in L can be simulated by L-SPCU queries. The data complexity of MDatalog in $KD4I_g5_a$ and *sCFG* is PSPACE-hard, and some operator of the L-SPCU algebra may return an infinite relation even when the input consists of finite relations. In Section 7, we overcome these problems by providing an approximation method for evaluating MDatalog queries in $KD4I_g5_a$ and *sCFG*. In Sections 8, 9, and 10, we present the seminaive evaluation algorithm, the top-down evaluation algorithm, and the magic-set transformation for MDatalog, respectively. We conclude this work in Section 11.

2 Preliminaries

2.1 Definitions for Quantified Multimodal Logics

A language for quantified multimodal logics is an extension of the language of classical predicate logic with modal operators $\Box_i$ and $\Diamond_i$, for $1 \leq i \leq m$ (where m is fixed). The modal operators $\Box_i$ and $\Diamond_i$ can take various meanings. For example, $\Box_i$ can stand for "the agent i believes" and $\Diamond_i$ for "it is considered possible by agent i". The operators $\Box_i$ are called universal modal operators, while $\Diamond_i$ are called existential modal operators. Terms and formulas are defined in the usual way, with an emphasis that if φ is a formula then $\Box_i\varphi$ and $\Diamond_i\varphi$ are also formulas.

A term or a formula is *ground* if it does not contain variables.

If φ is a formula, then 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[1] in φ. Similarly, $\exists(\varphi)$ denotes the *existential closure* of φ, which is obtained by adding an existential quantifier for every variable having a free occurrence in φ.

The *modal depth* of a formula φ, denoted by $mdepth(\varphi)$, is the maximal nesting depth of modal operators occurring in φ. For example, the modal depth of $\Box_i(\Diamond_j p(x) \vee \Box_k q(y))$ is 2.

We now define Kripke models, model graphs, and the satisfaction relation.

Definition 2.1 A *Kripke frame* is a tuple $\langle W, \tau, R_1, \ldots, R_m \rangle$, where W is a nonempty set of possible worlds, $\tau \in W$ is the *actual world*, and R_i is a binary relation on W, called the *accessibility relation* for the modal operators $\Box_i$, $\Diamond_i$. If $R_i(w, u)$ holds then we say that the world u is accessible from the world w via R_i.

A frame $\langle W, \tau, R_1, \ldots, R_m \rangle$ is said to be *connected* if each of its worlds is directly or indirectly accessible from the actual world via the accessibility relations, i.e. for every $w \in W$ there exist $w_0 = \tau, w_1, \ldots, w_{k-1}, w_k = w$ with $k \geq 0$ such that $(w_i, w_{i+1}) \in R_1 \cup \ldots \cup R_m$ for all $0 \leq i < k$.

Definition 2.2 A *fixed-domain Kripke model with rigid terms*, hereafter simply called a Kripke model or just a model, is a tuple $M = \langle D, W, \tau, R_1, \ldots, R_m, \pi \rangle$, where D is a set called the *domain*, $\langle W, \tau, R_1, \ldots, R_m \rangle$ is a Kripke frame, and π is an interpretation of constant symbols, function symbols and predicate symbols. For a constant symbol a, $\pi(a)$ is an element of D, denoted by a^M. For an n-ary function symbol f, $\pi(f)$ is a function from D^n to D, denoted by f^M. For an n-ary predicate symbol p and a world $w \in W$, $\pi(w)(p)$ is an n-ary relation on D, denoted by $p^{M,w}$.

[1] i.e. an occurrence not bound by quantifiers

Definition 2.3 A *model graph* is a tuple $\langle W, \tau, R_1, \ldots, R_m, H\rangle$, where $\langle W, \tau, R_1, \ldots, R_m\rangle$ is a Kripke frame and H is a function that maps each world of W to a set of formulas.

Every model graph $\langle W, \tau, R_1, \ldots, R_m, H\rangle$ corresponds to an Herbrand model $M = \langle \mathcal{U}, W, \tau, R_1, \ldots, R_m, \pi\rangle$ specified by: $\mathcal{U}$ is the Herbrand universe (i.e. the set of all ground terms), $c^M = c$, $f^M(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$, and $((t_1, \ldots, t_n) \in p^{M,w}) \equiv (p(t_1, \ldots, t_n) \in H(w))$, where $t_1, \ldots, t_n$ are ground terms. We will sometimes treat a model graph as its corresponding model.

Definition 2.4 Let M be a Kripke model. A *variable assignment* (w.r.t. M) is a function that maps each variable to an element of the domain of M. The value of a term t w.r.t. a variable assignment V is denoted by $t^M[V]$ and defined as follows: If t is a constant symbol a then $t^M[V] = a^M$; if t is a variable x then $t^M[V] = V(x)$; if t is $f(t_1, \ldots, t_n)$ then $t^M[V] = f^M(t_1^M[V], \ldots, t_n^M[V])$.

Definition 2.5 Given some Kripke model $M = \langle D, W, \tau, R_1, \ldots, R_m, \pi\rangle$, some variable assignment V, and some world $w \in W$, the *satisfaction relation* $M, V, w \vDash \psi$ for a formula ψ is defined as follows:

$$\begin{array}{lll}
M, V, w \vDash p(t_1, \ldots, t_n) & \text{iff} & (t_1^M[V], \ldots, t_n^M[V]) \in p^{M,w}; \\
M, V, w \vDash \Box_i \varphi & \text{iff} & \text{for all } v \in W \text{ such that } R_i(w, v),\ M, V, v \vDash \varphi; \\
M, V, w \vDash \forall x.\varphi & \text{iff} & \text{for all } a \in D,\ (M, V', w \vDash \varphi), \\
 & & \text{where } V'(x) = a \text{ and } V'(y) = V(y) \text{ for } y \neq x;
\end{array}$$

and as usual for other cases (treating $\Diamond_i \varphi$ as $\neg\Box_i\neg\varphi$, and $\exists x.\varphi$ as $\neg\forall x.\neg\varphi$). If $M, V, w \vDash \varphi$ then we say that φ is true at w in M w.r.t. V. We write $M, w \vDash \varphi$ to denote that $M, V, w \vDash \varphi$ for every V. We say that M satisfies φ, or φ is true in M, and write $M \vDash \varphi$, if $M, \tau \vDash \varphi$. For a set Γ of formulas, we call M a model of Γ and write $M \vDash \Gamma$ if $M \vDash \varphi$ for every $\varphi \in \Gamma$.

Let us explain why we include the actual world in the definition of Kripke models. Consider possible definitions of $M \vDash \Gamma$. Without the actual world, one would define that $M \vDash \Gamma$ if $M, w \vDash \Gamma$ for every world w of M. This is not appropriate for our settings of modal logic programming: for example, when Γ is a logic program containing a classical fact $p(a)$, then we do not require that $p(a)$ is true at every possible world of M, because otherwise it would imply that $p(a)$ is "known" to be true in M.

If the class of admissible interpretations contains all Kripke models (with no restrictions on the accessibility relations) then we obtain a quantified multimodal logic which has a standard Hilbert-style axiomatization denoted by $K_{(m)}$. Other *normal (multi)modal logics* are obtained by adding certain axioms to $K_{(m)}$. Mostly used axioms are ones that correspond to a certain restriction on the Kripke frame defined by a classical first-order formula using the accessibility relations. For example, the axiom $(D) : \Box_i\varphi \to \Diamond_i\varphi$ corresponds to the frame restriction $\forall x\, \exists y\, R_i(x, y)$. Normal modal logics containing this axiom (for all $1 \leq i \leq m$) are called *serial* modal logics.

For a normal modal logic L whose class of admissible interpretations can be characterized by classical first-order formulas of the accessibility relations, we call such formulas *L-frame restrictions*, and call frames with such properties *L-frames.*

Definition 2.6 We call a model M with an L-frame an *L-model.* We say that φ is *L-satisfiable* if there exists an L-model of φ, i.e. an L-model satisfying φ. A formula φ is said to be *L-valid* and called an *L-tautology* if φ is true in every L-model. For a set Γ of formulas, we write $\Gamma \vDash_L \varphi$ and call φ a *logical consequence* of Γ in L if φ is true in every L-model of Γ.

Note that our definition of $\Gamma \vDash_L \varphi$ reflects "local semantic consequence" due to the inclusion of actual world . Also note that $\Gamma \vDash_L \varphi$ means $\forall(\Gamma) \to \forall(\varphi)$ is an L-tautology.

2.2 Multimodal Logics of Belief

To reflect properties of belief, one can extend $K_{(m)}$ with some of the following axioms:

Name	Schema	Meaning
(D)	$\Box_i\varphi \to \neg\Box_i\neg\varphi$	belief is consistent
(I)	$\Box_i\varphi \to \Box_j\varphi$ if $i > j$	subscript indicates degree of belief
(4)	$\Box_i\varphi \to \Box_i\Box_i\varphi$	belief satisfies positive introspection
(4_s)	$\Box_i\varphi \to \Box_j\Box_i\varphi$	belief satisfies strong positive introspection
(5)	$\neg\Box_i\varphi \to \Box_i\neg\Box_i\varphi$	belief satisfies negative introspection
(5_s)	$\neg\Box_i\varphi \to \Box_j\neg\Box_i\varphi$	belief satisfies strong negative introspection

The following systems are intended for reasoning about multi-degree belief:

$$\begin{aligned} KDI4_s5 &= K_{(m)} + (D) + (I) + (4_s) + (5) \\ KDI45 &= K_{(m)} + (D) + (I) + (4) + (5) \end{aligned}$$

In the above systems, the axiom (I) gives $\Box_i\varphi$ the meaning "φ is believed up to degree i", and $\Diamond_i\varphi$ can be read as "it is possible weakly at degree i that φ". Note that the axiom (5_s) is derivable in $KDI4_s5$.

For multi-agent systems, we use subscripts beside $\Box$ and $\Diamond$ to denote agents and assume that $\Box_i\varphi$ stands for "agent i believes that φ is true" and $\Diamond_i\varphi$ stands for "φ is considered possible by agent i". For distributed systems of belief we can use the logic system

$$KD4_s5_s = K_{(m)} + (D) + (4_s) + (5_s)$$

In this system, agents have full access to belief bases of each other. They are "friends" in a united system. In another kind of multi-agent system, agents are "opponents" and they play against each other. Each one of the agents may want to simulate epistemic states of the others. To write a program for an agent, one may need to use modal operators of the other agents. A suitable logic for this problem is:

$$KD45_{(m)} = K_{(m)} + (D) + (4) + (5)$$

We use a subscript in $KD45_{(m)}$ to distinguish the logic from the monomodal logic $KD45$, while there is not such a need for the other considered multimodal logics.

To capture common belief of a group of agents, one can extend the logic $KD45_{(m)}$ with modal operators for groups of agents and some additional axioms. Suppose that there are n agents and $m = 2^n - 1$. Let g be an one-to-one function that maps every natural number less than or equal to m to a non-empty subset of $\{1, \dots, n\}$. Suppose that an index $1 \le i \le m$ stands for the group of agents whose indices form the set $g(i)$. (We want to use $1, \dots, m$ as modal indices for all the considered modal logics and we need the function g for comparing groups of agents.) We can adopt the axioms (D), (4), and additionally,

$$\begin{array}{ll} (I_g) : \Box_i\varphi \to \Box_j\varphi & \text{if } g(i) \supset g(j) \\ (5_a) : \neg\Box_i\varphi \to \Box_i\neg\Box_i\varphi & \text{if } g(i) \text{ is a singleton} \end{array}$$

The condition of (I_g) states that i indicates a group containing the group identified by j, and the condition of (5_a) states that i stands for an agent. Thus, for reasoning about belief and common belief, we can use:

$$KD4I_g5_a = K_{(m)} + (D) + (4) + (I_g) + (5_a)$$

Here we want to catch the most important properties of belief and common belief, and the aim is not to give an exact formulation of belief or common belief. This logic is different in the nature from the well-known multimodal logic of common knowledge. It also differs from the multimodal logic of mutual belief introduced by Aldewereld *et al.* [2]. Our modal operator of common belief satisfies positive introspection, while the operator of mutual belief introduced in [2] lacks this property. On the other hand, the latter operator has some properties that the former does not have.

The given axioms correspond to the following frame restrictions:

Axiom	Corresponding Condition
(D)	$\forall u\ \exists v\ R_i(u,v)$
(I)	$R_j \subseteq R_i$ if $i > j$
(I_g)	$R_j \subseteq R_i$ if $g(i) \supseteq g(j)$
(4)	$\forall u,v,w\ (R_i(u,v) \wedge R_i(v,w) \to R_i(u,w))$
(4_s)	$\forall u,v,w\ (R_j(u,v) \wedge R_i(v,w) \to R_i(u,w))$
(5)	$\forall u,v,w\ (R_i(u,v) \wedge R_i(u,w) \to R_i(w,v))$
(5_s)	$\forall u,v,w\ (R_j(u,v) \wedge R_i(u,w) \to R_i(v,w))$
(5_a)	as for (5) if $g(i)$ is a singleton

As an example, it can be checked that a connected frame $\langle W, \tau, R_1, \ldots, R_m\rangle$ is a $KDI4_s5$-frame iff there are nonempty subsets of worlds $W_1 \subseteq \ldots \subseteq W_m$ such that $W = \{\tau\} \cup W_m$ and $R_i = W \times W_i$, for $1 \leq i \leq m$.

2.3 Serial Context-Free Grammar Logics

A *grammar logic* is a multimodal logic extending $K_{(m)}$ with "inclusion axioms" of the form $\Box_{i_1} \ldots \Box_{i_k}\varphi \to \Box_{j_1} \ldots \Box_{j_h}\varphi$. Such logics were introduced by Fariñas del Cerro and Penttonen in [8]. An inclusion axiom $\Box_{i_1} \ldots \Box_{i_k}\varphi \to \Box_{j_1} \ldots \Box_{j_h}\varphi$ corresponds to the restriction $R_{j_1} \circ \ldots \circ R_{j_h} \subseteq R_{i_1} \circ \ldots \circ R_{i_k}$ on accessibility relations. If $k = 0$ (resp. $h = 0$) then the LHS (resp. RHS) of the inclusion stands for the identity relation.

An inclusion axiom $\Box_{i_1} \ldots \Box_{i_k}\varphi \to \Box_{j_1} \ldots \Box_{j_h}\varphi$ can also be seen as the grammar rule $i_1 \ldots i_k \to j_1 \ldots j_h$ where, if $k = 0$ or $h = 0$ then the corresponding side stands for the empty word. Thus the inclusion axioms of a grammar logic L capture a grammar $\mathcal{G}(L)$. Here we do not distinguish terminal symbols and nonterminal symbols. $\mathcal{G}(L)$ is *context-free* if its rules are of the form $i \to j_1 \ldots j_k$.

A *context-free grammar logic* L is a grammar logic whose inclusion axioms correspond to grammar rules that collectively capture a context-free grammar $\mathcal{G}(L)$. A *serial context-free grammar logic* (*sCFG logic* for short) is an extension of a context-free grammar logic with the axiom $(D) : \Box_i\varphi \to \Diamond_i\varphi$ (for every $1 \leq i \leq m$).

Proposition 2.1 *Let L be an sCFG logic. Then the following conditions are equivalent:*

1. $\Box_{i_1} \ldots \Box_{i_k}\varphi \to \Box_{j_1} \ldots \Box_{j_h}\varphi$ *is L-valid for any φ.*
2. $\Box_{i_1} \ldots \Box_{i_k}\varphi \to \Box_{j_1} \ldots \Box_{j_h}\varphi$ *is derivable in L for any φ without using axiom (D).*
3. $j_1 \ldots j_h$ *is derivable from $i_1 \ldots i_k$ using the context-free grammar $\mathcal{G}(L)$.*

See [27] for the proof of this proposition.

Corollary 2.2 *Let L be an sCFG logic, $\boxdot$ and $\boxdot'$ be universal modalities. Then the problem of checking whether $\boxdot'\varphi \to \boxdot\varphi$ is L-valid for any φ is decidable.*

Proof. This corollary follows from Proposition 2.1 and the fact that the derivation problem in context-free grammars is decidable. •

We sometimes use *sCFG* also to denote an arbitrary logic belonging to the *sCFG* class.

For further reading on modal logics, we refer the reader to [6, 9, 3, 11].

2.4 Ordering Kripke Models

A formula is in *negation normal form* if it does not contain the connective $\to$ and in which each negation occurs immediately before a classical atom. Every formula can be transformed to its equivalent negation normal form in the usual way. A formula is called *positive* if its negation normal form does not contain negation. A formula is called *negative* if its negation is a positive formula.

Definition 2.7 A model M is said to be *less than* or *equal to* N, write $M \leq N$, if for any positive ground formula φ, if M satisfies φ then N also satisfies φ.

The relation $\leq$ in the above definition is a pre-order[2]. It is defined semantically and is not easy to be checked. For checking, we can use the following syntactic ordering.

Definition 2.8 Let $M = \langle D, W, \tau, R_1, \ldots, R_m, \pi\rangle$ and $N = \langle D', W', \tau', R'_1, \ldots, R'_m, \pi'\rangle$ be Kripke models. We say that M is *less than or equal to N w.r.t. a binary relation $r \subseteq W \times W'$*, and write $M \leq_r N$, if the following conditions hold:

1. $r(\tau, \tau')$.
2. $\forall x, x', y \;\; R_i(x, y) \wedge r(x, x') \to \exists y' \; R'_i(x', y') \wedge r(y, y')$, for all $1 \leq i \leq m$.
3. $\forall x, x', y' \;\; R'_i(x', y') \wedge r(x, x') \to \exists y \; R_i(x, y) \wedge r(y, y')$, for all $1 \leq i \leq m$.
4. For any $x \in W$ and $x' \in W'$ such that $r(x, x')$, and for any ground classical atom E, if $M, x \vDash E$ then $N, x' \vDash E$.

In the above definition, the first three conditions state that r is a bisimulation of the frames of M and N. Intuitively, $r(x, x')$ states that the world x is "less than or equal" to x' (i.e. for every positive ground formula φ, if $M, x \vDash \varphi$ then $N, x' \vDash \varphi$).

Lemma 2.3 *If $M \leq_r N$ then $M \leq N$.*

This lemma can be proved by induction on the length of φ that, if φ is a positive ground formula and $r(x, x')$ holds then $M, x \vDash \varphi$ implies $N, x' \vDash \varphi$ (cf. [19]).

2.5 Positive Multimodal Logic Programs

A *modality* is a (possibly empty) sequence of modal operators. A *universal modality* is a modality that contains only universal modal operators. We use $\triangle$ to denote a modality and $\boxdot$ to denote a universal modality. Similarly as in classical logic programming, we use a clausal form $\boxdot(\varphi \leftarrow \psi_1, \ldots, \psi_n)$ to denote the formula $\forall(\boxdot(\varphi \vee \neg\psi_1 \ldots \vee \neg\psi_n))$. We use E to denote a classical atom.

Definition 2.9 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, $\ldots$, B_n are formulas of the form E, $\Box_i E$, or $\Diamond_i E$ with E being a classical atom. $\boxdot$ is called the *modal context*, A the *head*, and $B_1, \ldots, B_n$ the *body* of the program clause. If $n = 0$ then we call the clause a *unary* clause.

Definition 2.10 An *MProlog program* is a finite set of program clauses.

Definition 2.11 An *MProlog goal atom* is a formula of the form $\boxdot E$ or $\boxdot\Diamond_i E$, where E is a classical atom. An *MProlog query* is a formula of the form $\exists(\alpha_1 \wedge \ldots \wedge \alpha_k)$, where α_1, $\ldots$, α_k are MProlog goal atoms. An *MProlog goal* is the negation of an MProlog query, written in the form $\leftarrow \alpha_1, \ldots, \alpha_k$. We denote the *empty goal* (also called the *empty clause*) by $\diamond$.

If P is an MProlog program, $Q = \exists(\alpha_1 \wedge \ldots \wedge \alpha_k)$ is an MProlog query and $G = \leftarrow \alpha_1, \ldots, \alpha_k$ is the corresponding goal, then $P \vDash_L Q$ iff $P \cup \{G\}$ is L-unsatisfiable. For the proof of this statement, just note that $G = \forall(\neg(\alpha_1 \wedge \ldots \wedge \alpha_k))$.

When the base logic is intended for reasoning about multi-degree belief, it has little sense to write a program clause in the form $\Box_i\Box_j\varphi$ or a goal in the form $\leftarrow \Box_i\Box_j E$ or $\leftarrow \Box_i\Diamond_j E$. Besides, in the logics $KDI4_s5$ and $KD4_s5_s$ we have the tautology $\nabla\nabla'\varphi \equiv \nabla'\varphi$, where ∇ and ∇' denote modal operators. For these reasons, we introduce some restrictions for MProlog programs and goals in these logics.

Definition 2.12 For $L \in \{KDI4_s5, KDI45, KD4_s5_s\}$, an MProlog program is called an *L-MProlog program* if its program clauses have modal contexts with length 0 or 1, an MProlog goal is called an *L-MProlog goal* if its modal depth is 0 or 1. (Recall that the modal depth of φ is the maximal nesting depth of modal operators occurring in φ.)

[2]i.e. a reflexive and transitive binary relation

In the logic $KD45_{(m)}$, we have the tautologies $\Box_i\Box_i\varphi \equiv \Box_i\varphi$ and $\Box_i\Diamond_i\varphi \equiv \Diamond_i\varphi$. In $KD4I_g5_a$, these two equivalences hold for the case when $g(i)$ is a singleton. So, we introduce restrictions for MProlog programs and goals in $KD45_{(m)}$ and $KD4I_g5_a$.

Definition 2.13 An MProlog program is called a $KD45_{(m)}$*-MProlog program* if the modal contexts of its program clauses do not contain subsequences of the form $\Box_i\Box_i$. An MProlog goal is called a $KD45_{(m)}$*-MProlog goal* if each of its goal atoms $\triangle E$ satisfies the condition that $\triangle$ does not contain subsequences of the form $\Box_i\Box_i$ or $\Box_i\Diamond_i$. $KD4I_g5_a$*-MProlog* programs and goals are defined similarly with the condition that $g(i)$ is a singleton.

For L not mentioned in the two above definitions (e.g. $L = sCFG$), assume that no restriction is adopted for the form of L-MProlog programs and goals (i.e. every MProlog program is an L-MProlog program, and every MProlog goal is an L-MProlog goal).

Definition 2.14 Let P be an L-MProlog program and $G = \leftarrow \alpha_1, \ldots, \alpha_k$ be an L-MProlog goal. An *answer* θ for $P \cup \{G\}$ is a substitution whose domain is a set of variables of G. We say that θ is a *correct answer* in L for $P \cup \{G\}$ if θ is an answer for $P \cup \{G\}$ and $P \models_L \forall((\alpha_1 \wedge \ldots \wedge \alpha_k)\theta)$.

It is shown in [27] that MProlog has the same expressiveness power as the general Horn fragment in normal modal logics. Furthermore, *the restrictions adopted for L-MProlog do not reduce expressiveness of the language* (see [27]).

Example 2.1 Let us consider the situation when a company has some branches and a central database. Each of the branches can access and update the database, and suppose that the company wants to distinguish data and knowledge coming from different branches. Also assume that data coming from branches can contain noises and statements expressed by a branch may not be highly recognized by other branches. This means that data and statements expressed by branches are treated as "belief" rather than "knowledge". In this case, we can use the multimodal logic $KD4_s5_s$, where each modal index represents a branch of the company, also called an *agent*. Recall that in this logic each agent has full access to the belief bases of the other agents. Data put by agent i are of the form $\Box_i E$ (agent i believes in E) or $\Diamond_i E$ (agent i considers that E is possible). A statement expressed by agent i is a clause of the form $\Box_i(A \leftarrow B_1, \ldots, B_n)$, where A is an atom of the form E, $\Box_i E$, or $\Diamond_i E$, and $B_1, \ldots, B_n$ are simple modal atoms that may contain modal operators of the other agents. For communicating with normal users, the central database may contain clauses with the empty modal context, i.e. in the form $E \leftarrow B_1, \ldots, B_n$, which hide sources of information. As a concrete example, consider the following program/database P_{ddb} in $KD4_s5_s$:

agent 1:
$\varphi_1 = \Box_1 likes(Jan, cola) \leftarrow$
$\varphi_2 = \Box_1 likes(Piotr, pepsi) \leftarrow$
$\varphi_3 = \Box_1(\Diamond_1 likes(x, cola) \leftarrow likes(x, pepsi))$
$\varphi_4 = \Box_1(\Diamond_1 likes(x, pepsi) \leftarrow likes(x, cola))$
agent 2:
$\varphi_5 = \Box_2 likes(Jan, pepsi) \leftarrow$
$\varphi_6 = \Box_2 likes(Piotr, cola) \leftarrow$
$\varphi_7 = \Box_2 likes(Piotr, beer) \leftarrow$
$\varphi_8 = \Box_2(likes(x, cola) \leftarrow likes(x, pepsi))$
$\varphi_9 = \Box_2(likes(x, pepsi) \leftarrow likes(x, cola))$
agent 3:
$\varphi_{10} = \Box_3 likes(Jan, cola) \leftarrow$
$\varphi_{11} = \Diamond_3 likes(Piotr, pepsi) \leftarrow$
$\varphi_{12} = \Diamond_3 likes(Piotr, beer) \leftarrow$
$\varphi_{13} = \Box_3(very\_much\_likes(x, y) \leftarrow likes(x, y), \Box_1 likes(x, y), \Box_2 likes(x, y))$
agent communicating with users:
$\varphi_{14} = very\_much\_likes(x, y) \leftarrow \Box_3 very\_much\_likes(x, y)$
$\varphi_{15} = likes(x, y) \leftarrow \Diamond_3 very\_much\_likes(x, y)$
$\varphi_{16} = possibly\_likes(x, y) \leftarrow \Diamond_i likes(x, y)$

The modal index i in φ_{16} can take value 1, 2, or 3. Let the base logic be $KD4_s5_s$. For the goal $\leftarrow very\_much\_likes(x, y)$, we have the unique correct answer $\{x/Jan, y/cola\}$. For the goal $\leftarrow likes(x, y)$, we have two correct answers $\{x/Jan, y/cola\}$ and $\{x/Piotr, y/pepsi\}$. For the goal $\leftarrow possibly\_likes(x, y)$, we have 5 correct answers.

3 A Framework for Multimodal Logic Programming

In this section, we briefly recall our framework given in [25] for developing least model semantics, fixpoint semantics and SLD-resolution calculi for L-MProlog programs. For an illustrating example, we refer the reader to the Introduction of [25]. The base logic L is required to be a normal multimodal logic such that the set of L-frame restrictions consists of $\forall x\, \exists y\, R_i(x, y)$ (seriality), for all $1 \leq i \leq m$, and some classical first-order Horn clauses. The restriction of seriality is to guarantee the existence of least models of MProlog programs. It is also needed for our fixpoint semantics and SLD-resolution calculi for MProlog, because they are based on the assumption that $\Diamond_i$ is an "instance" of $\Box_i$. In this section, we also present instantiations of the framework for the multimodal logics specified in Sections 2.2 and 2.3.

3.1 Labeled Modal Operators and Notations

In classical logic programming, the direct consequence operator T_P acts on sets of ground atoms. It computes "direct" consequences of the input set using the program clauses of P. The operator is monotonic and continuous and has the least fixpoint, which is a set of atoms forming the least Herbrand model of P. In modal logic programming, to obtain a similar result we first have to decide what is the domain of the direct consequence operator $T_{L,P}$. Naturally, we still want it to be the class of sets of *atoms*. But what is an "atom" in this case? When applying the operator $T_{L,P}$ to an input set, if we obtain some atom of the form $\triangle\Diamond_i E$ (where $\triangle$ is a modality and E is a classical atom), then to simplify the computation we label the modal operator $\Diamond_i$ to address the chosen world(s) in which this particular E must hold. A natural way is to label $\Diamond_i$ by E to obtain $\langle E\rangle_i$. Thus, an output/input of $T_{L,P}$ consists of atoms of the form $\triangle E$, where $\triangle$ is a sequence of modal operators of the form $\Box_i$ or $\langle F\rangle_i$, with E, F being ground classical atoms.

On the other hand, when dealing with SLD-derivation, we cannot change a goal $\leftarrow \Diamond_i(A \wedge B)$ to $\leftarrow \Diamond_i A, \Diamond_i B$ because this is not equivalence preserving (as the two existential modal operators can refer to different possible worlds). But if we label the operator $\Diamond_i$, let's say by X, so that the existential modal operators will refer to the same possible world, then we can safely change $\leftarrow \langle X\rangle_i(A \wedge B)$ to $\leftarrow \langle X\rangle_i A, \langle X\rangle_i B$. Such a variable X may later be unified with a classical atom, so we call it an *atom variable*.

We will use the following notations:

- $\top$: the *truth* symbol, with the usual semantics[3];
- E, F : classical atoms (which may contain variables) or $\top$;
- X, Y, Z : variables for classical atoms or $\top$, called *atom variables*;
- $\langle E\rangle_i$, $\langle X\rangle_i$: $\Diamond_i$ labeled by E or X;
- ∇ : $\Box_i$, $\Diamond_i$, $\langle E\rangle_i$, or $\langle X\rangle_i$, called a modal operator;
- $\triangle$: a (possibly empty) sequence of modal operators, called a *modality*;
- $\boxdot$: a *universal modality* (i.e. a modality containing only universal modal operators);
- A, B : formulas of the form E or ∇E, called *simple atoms*;
- α, β : formulas of the form $\triangle E$, called *atoms*;
- φ, ψ : *(labeled) formulas* (i.e. formulas that may contain $\langle E\rangle_i$ and $\langle X\rangle_i$).

[3]i.e. it is always true that $M, V, w \vDash \top$

We use subscripts beside ∇ to indicate modal indices in the same way as for $\Box$ and $\Diamond$. To distinguish a number of modal operators we use superscripts, e.g. ∇', $\nabla^{(i)}$, $\nabla^{(i')}$.

If a modality $\triangle$ is obtainable from $\triangle'$ by replacing some (possibly zero) ∇_i by $\Box_i$ then we call $\triangle$ a *$\Box$-lifting form* of $\triangle'$. If $\triangle$ is a $\Box$-lifting form of $\triangle'$ then we call an atom $\triangle\alpha$ a *$\Box$-lifting form* of $\triangle'\alpha$. For example, $\Box_1\langle p(a)\rangle_1\Box_2 q(b)$ is a $\Box$-lifting form of $\langle X\rangle_1\langle p(a)\rangle_1\Diamond_2 q(b)$.

A *ground formula* is redefined to be a formula with no variables and no atom variables. A modal operator is said to be *ground* if it is $\Box_i$, $\Diamond_i$, or $\langle E\rangle_i$ with E being $\top$ or a ground classical atom. A *ground modality* is a modality that contains only ground modal operators. A *labeled modal operator* is a modal operator of the form $\langle E\rangle_i$ or $\langle X\rangle_i$.

We redefine also substitutions in order to deal with atom variables and labeled formulas. Definitions involving with substitution and unification change accordingly in the usual way.

Definition 3.1 A *substitution* θ is a (finite or infinite) set of the form $\{x_1/t_1, x_2/t_2, \ldots, X_1/E_1, X_2/E_2, \ldots, Y_1/Z_1, Y_2/Z_2, \ldots\}$, where $x_1, x_2, \ldots$ are distinct variables, $t_1, t_2, \ldots$ are terms, $X_1, X_2, \ldots, Y_1, Y_2, \ldots$ are distinct atom variables, and for any element v/s of the set, s is distinct from v. The set $\{x_1, x_2, \ldots, X_1, X_2, \ldots, Y_1, Y_2, \ldots\}$ is called the *domain* of θ and denoted by $Dom(\theta)$. A substitution θ is said to be *ground* if the set $\{Y_1, Y_2, \ldots\}$ is empty, $t_1, t_2, \ldots$ are ground terms, and $E_1, E_2, \ldots$ are ground classical atoms.

Denote $EdgeLabels = \{\langle E\rangle_i \mid E \in \mathcal{B} \cup \{\top\} \text{ and } 1 \leq i \leq m\}$, where $\mathcal{B}$ is the Herbrand base (i.e. the set of all ground classical atoms). The semantics of $\langle E\rangle_i \in EdgeLabels$ is specified below.

Definition 3.2 Let $M = \langle D, W, \tau, R_1, \ldots, R_m, \pi\rangle$ be a Kripke model. A *$\Diamond$-realization function on M* is a partial function $\sigma : W \times EdgeLabels \to W$ such that if $\sigma(w, \langle E\rangle_i) = u$, then $R_i(w, u)$ holds and $M, u \vDash E$. Given a $\Diamond$-realization function σ, a world $w \in W$, and a ground formula φ, the satisfaction relation $M, \sigma, w \vDash \varphi$ is defined in the usual way, except that $M, \sigma, w \vDash \langle E\rangle_i\psi$ iff $\sigma(w, \langle E\rangle_i)$ is defined and $M, \sigma, \sigma(w, \langle E\rangle_i) \vDash \psi$. We write $M, \sigma \vDash \varphi$ to denote that $M, \sigma, \tau \vDash \varphi$. For a set I of ground atoms, we write $M, \sigma \vDash I$ to denote that $M, \sigma \vDash \alpha$ for all $\alpha \in I$; we write $M \vDash I$ and call M a model of I if $M, \sigma \vDash I$ for *some* σ.

Definition 3.3 Let σ and σ' be $\Diamond$-realization functions on a model M. We say that σ is an *extension* of σ' if whenever $\sigma'(w, \langle E\rangle_i)$ is defined then $\sigma(w, \langle E\rangle_i) = \sigma'(w, \langle E\rangle_i)$. We say that σ is a *maximal* $\Diamond$-realization function on M if $\sigma(w, \langle E\rangle_i)$ is defined whenever $M, w \vDash \Diamond_i E$.

Atom variables in modal operators of the form $\langle X\rangle_i$ are mainly interpreted by substitutions. When a formula φ is taken to be semantically considered, all modal operators $\langle X\rangle_i$ in φ are treated as[4] $\langle\top\rangle_i$, which is formalized by the following definition.

Definition 3.4 Given a Kripke model M, a $\Diamond$-realization function σ, and a labeled formula φ without quantifiers, we write $M, \sigma \vDash \forall_c(\varphi)$ to denote that for any substitution θ which substitutes every variable by a ground term and does not substitute atom variables, $M, \sigma \vDash \varphi\theta\delta_\top$, where $\delta_\top = \{X/\top \mid X \text{ is an atom variable}\}$. By $M \vDash \forall_c(\varphi)$ we denote $M, \sigma \vDash \forall_c(\varphi)$ for *some* σ.

If Γ is a set of formulas without labeled modal operators, I is a set of ground atoms, and φ is a formula without quantifiers, then the relations $\Gamma \vDash_L I$ and $\Gamma \vDash_L \forall_c(\varphi)$ are interpreted as usual.

The quantifier $\forall_c$ is introduced because $\Diamond$-realization functions are defined using Herbrand base and we do not want to restrict only to Herbrand models. Suppose that there are enough constant symbols not occurring in Γ, for example, infinitely many. Then, because L has a complete axiomatization, for Γ being a formula set and φ a formula – both without labeled modal operators, $\Gamma \vDash_L \forall(\varphi)$ iff $\Gamma \vDash_L \forall_c(\varphi)$.

[4]Atom variables appear only in *goal* bodies (see Definition 2.11). In the negation of a goal (i.e. a query) they are existentially quantified. Hence it is sufficient to choose some concrete values for them. Furthermore, as we will see, the modal operator $\langle\top\rangle_i$ plays the role of $\Box_i$; and if X remains at the end as an unsubstituted atom variable then $\langle X\rangle_i$ intuitively also plays the role of $\Box_i$.

3.2 Model Generators

As mentioned earlier, we will define the direct consequence operator $T_{L,P}$ for an L-MProlog program P so that an output/input of $T_{L,P}$ consists of atoms of the form $\triangle E$, where $\triangle$ is a sequence of modal operators of the form $\Box_i$ or $\langle F\rangle_i$, with E, F being ground classical atoms. For the reason that the least fixpoint of $T_{L,P}$ should represent a least L-model of P, we call inputs/outputs of $T_{L,P}$ *model generators.*

Definition 3.5 A *model generator* is a set of ground atoms not containing $\Diamond_i$, $\langle\top\rangle_i$, $\top$.

Because an atom in L may be reducible to some more compact form, for each specific logic L we will define *L-normal form of modalities.* It is possible that no restrictions on L-normal form of modalities are adopted.

Definition 3.6 For $L \in \{KDI4_s5, KD4_s5_s, KD45_{(m)}, KDI45, KD4I_g5_a, sCFG\}$, a modality $\nabla^{(1)}_{i_1} \ldots \nabla^{(k)}_{i_k}$ is in L-normal form if

- case $L \in \{KDI4_s5, KD4_s5_s\}$: $k \leq 1$,
- case $L = KD45_{(m)}$: $i_j \neq i_{j+1}$ for all $1 \leq j < k$,
- case $L = KDI45$: $i_1 > \ldots > i_k$,
- case $L = KD4I_g5_a$: for all $1 \leq j < k$, if $g(i_j)$ is a singleton then $i_j \neq i_{j+1}$,
- case $L = sCFG$: no restrictions.

Definition 3.7 A modality $\triangle$ is in *L-normal labeled form* if it is in L-normal form and does not contain modal operators of the form $\Diamond_i$ or $\langle\top\rangle_i$. An atom is in *L-normal (labeled) form* if it is of the form $\triangle E$ with $\triangle$ in L-normal (labeled) form. (Recall that E denotes a classical atom or $\top$.) An atom is in *almost L-normal labeled form* if it is of the form $\triangle A$ with $\triangle$ in L-normal labeled form. (Recall that A denotes a simple atom of the form E or ∇E, where ∇ is a modal operator possibly not labeled.)

Let $F \neq \top$ in this example. The modalities $\Box_i$ and $\langle F\rangle_i$ are in $KDI4_s5$-normal labeled form, while $\Box_i\Box_j$, $\Diamond_i$, $\langle\top\rangle_i$ are not. Atoms E, $\Box_i E$, $\langle F\rangle_i E$ are in $KDI4_s5$-normal labeled form, while $\Box_i\Box_j E$, $\Diamond_i E$, $\langle\top\rangle_i E$ are not. Atoms E, $\Box_i E$, $\Diamond_i E$, $\Box_i\Box_j E$, $\Box_i\Diamond_j E$, $\langle F\rangle_i E$ are in almost $KDI4_s5$-normal labeled form, while $\Diamond_i\Box_j E$ and $\Box_i\Box_j\Box_k E$ are not.

Definition 3.8 An *L-normal model generator* is a model generator consisting of atoms in L-normal form.

An L-normal model generator I is expected to represent an L-model. This specific model is called the *standard L-model* of I. It should contain only (positive) information that come from I. This means that the standard L-model of I should be a least L-model of I.

Given an L-normal model generator I, we can construct a least L-model for it by building an L-model graph realizing I. We identify possible worlds by finite sequences of ground labeled existential modal operators. The actual world is identified by ε (the empty sequence). Formulas of the form $\Box_i\alpha$ are realized in the usual way; a formula of the form $\langle E\rangle_i\alpha$ is realized at a world w by connecting w to a world identified by $w\langle E\rangle_i$ via R_i and adding α to that world. To guarantee the constructed model graph to be the smallest, each new world is connected via each R_i to an empty world at the time of its creation. Sometimes, the accessibility relations are extended to satisfy all of the L-frame restrictions.

We want to give here a more declarative definition of the standard L-model of an L-normal model generator I. The part specific to L is extracted into Ext_L and $Serial_L$, where $Ext_L(I)$ is an L-normal model generator extending I, and $Serial_L$ is a set of atoms of the form $\boxdot\langle\top\rangle_i\top$. The standard L-model of I is then defined using $Ext_L(I)$ and $Serial_L$ in a unified way, almost independently from L. The set $Serial_L$ is intended to guarantee that, for every world w and $1 \leq i \leq m$, w will be connected to a world which is "less than or equal to" every world accessible from w via R_i.

Definition 3.9 Define $Serial_L = \{\boxdot\langle\top\rangle_i\top \mid 1 \leq i \leq m$ and $\boxdot\langle\top\rangle_i$ is in L-normal form$\}$.

A *forward rule* is a schema of the form $\alpha \to \beta$, while a *backward rule* is a schema of the form $\alpha \leftarrow \beta$. (Recall that we use α and β to denote atoms, i.e. formulas of the form $\triangle E$.) A rule can be accompanied with some conditions specifying when the rule can be applied. We use forward rules to specify the operators Ext_L and Sat_L (needed for defining fixpoint semantics) and use backward rules as meta-clauses when dealing with SLD-resolution calculi. In practice, conditions for applying a backward rule can be attached to the body of the rule, and in general, a backward rule can be of the form $(\alpha \leftarrow \varphi, \beta, \psi)$ with φ and ψ being conjunctions of classical atoms. In this work, we just define that a backward rule is of the form $\alpha \leftarrow \beta$.

Definition 3.10 The *operator* Ext_L is specified by a finite set of forward rules. Given an L-normal model generator I, $Ext_L(I)$ is the least extension of I that contains all ground atoms in L-normal labeled form that are derivable from some atom of I using the rules specifying Ext_L.

Note that $Ext_L(I)$ is an L-normal model generator if so is I.

Definition 3.11 For $L \in \{KDI4_s5, KD4_s5_s, KD45_{(m)}, KDI45, KD4I_g5_a, sCFG\}$, Ext_L is specified by the following rules, in which formulas in both sides are required to be in L-normal labeled form for $L \notin \{KD4I_g5_a, sCFG\}$ (denote this restriction by (*)).

$L \in \{KD4_s5_s, KD45_{(m)}\}$:	no rules
$L = KDI4_s5$:	$\Box_i E \to \Box_j E$ if $i > j$
$L = KDI45$:	$\triangle\Box_i\alpha \to \triangle\Box_j\alpha$ if $i > j$
	$\triangle\Box_i\alpha \to \triangle\Box_i\Box_j\alpha$ if $i > j$
	$\triangle\Box_i\Box_j\alpha \to \triangle\Box_j\alpha$ if $i > j$
$L = KD4I_g5_a$:	$\triangle\Box_i\alpha \to \triangle\Box_j\alpha$ if $g(i) \supset g(j)$
	$\triangle\Box_i\alpha \to \triangle\Box_i\Box_i\alpha$
	$\triangle\nabla_i\Box_i\alpha \to \triangle\Box_i\alpha$ if $g(i)$ is a singleton
$L = sCFG$:	$\triangle\Box_i\alpha \to \triangle\Box_{j_1}\ldots\Box_{j_k}\alpha$ if $\Box_i\varphi \to \Box_{j_1}\ldots\Box_{j_k}\varphi$ is an axiom of L

As an example, for $L = KDI4_s5$, $Ext_L(\{\Box_2 E\}) = \{\Box_2 E, \Box_1 E\}$.

Definition 3.12 Let I be an L-normal model generator. The *standard L-model* of I is defined as follows. Let $W' = EdgeLabels^*$ (i.e. the set of all finite sequences of elements of $\{\langle E\rangle_i \mid E \in \mathcal{B} \cup \{\top\}$ and $1 \leq i \leq m\}$, where $\mathcal{B}$ is the Herbrand base), $\tau = \varepsilon$ (the empty sequence), $H(\tau) = Ext_L(I) \cup Serial_L$. Let $R'_i \subseteq W' \times W'$ and $H(u)$, for $u \in W'$, $u \neq \tau$, be the least sets such that:

- if $\langle E\rangle_i\alpha \in H(w)$, then $R'_i(w, w\langle E\rangle_i)$ holds and $\{E, \alpha\} \subseteq H(w\langle E\rangle_i)$;
- if $\Box_i\alpha \in H(w)$ and $R'_i(w, w\langle E\rangle_i)$ holds, then $\alpha \in H(w\langle E\rangle_i)$.

Let R_i, for $1 \leq i \leq m$, be the least extension of R'_i such that $(R_i)_{1\leq i\leq m}$ satisfies all the L-frame restrictions except seriality (which is cared by $Serial_L$)[5]. Let W be W' without worlds not accessible directly nor indirectly from τ via the accessibility relations R_i. We call the model graph $\langle W, \tau, R_1, \ldots, R_m, H\rangle$ the *standard L-model graph* of I, and its corresponding model M the *standard L-model* of I. $(R_i)_{1\leq i\leq m}$ is called the *skeleton* of M. By the *standard $\Diamond$-realization function on M* we call the $\Diamond$-realization function σ defined as follows: if $R'_i(w, w\langle E\rangle_i)$ holds then $\sigma(w, \langle E\rangle_i) = w\langle E\rangle_i$, else $\sigma(w, \langle E\rangle_i)$ is undefined.

[5]The least extension exists due to the assumption that all L-frame restrictions not concerning seriality are classical first-order Horn formulas.

Example 3.1 Let us give an example for the above construction. Consider the L-normal model generator $I = \{\langle p(a)\rangle_1 p(a), \Box_1 q(a), \Box_2 q(b)\}$ in $L = KDI4_s5$, with $m = 2$ (recall that m is the maximal modal index). We have $Ext_L(I) = I \cup \{\Box_1 q(b)\}$ (due to the rule $\Box_i E \to \Box_j E$ if $i > j$) and $Serial_L = \{\langle\top\rangle_1\top, \langle\top\rangle_2\top\}$. The standard L-model of I is specified as follows:

- $W = \{\tau, \langle p(a)\rangle_1, \langle\top\rangle_1, \langle\top\rangle_2\}$ is the set of possible worlds.
- τ is the actual world.
- $R_1 = W \times W_1$ and $R_2 = W \times W_2$ are the accessibility relations, where $W_1 = \{\langle p(a)\rangle_1, \langle\top\rangle_1\}$ and $W_2 = W_1 \cup \{\langle\top\rangle_2\}$.
- The world τ is empty; the world $\langle p(a)\rangle_1$ contains $p(a)$, $q(a)$, $q(b)$; the world $\langle\top\rangle_1$ contains $\top$, $q(a)$, $q(b)$; the world $\langle\top\rangle_2$ contains $\top$ and $q(b)$.

A model is a least L-model of an L-normal model generator I if it is an L-model of I and is less than or equal to every L-model of I.

We give below expected results about model generators. Their proofs for $L \in \{KDI4_s5$, $KD4_s5_s$, $KD45_{(m)}$, $KDI45$, $KD4I_g5_a$, $sCFG\}$ are given in [27].

Lemma 3.1 *Let I be an L-normal model generator, M the standard L-model of I, and σ the standard $\Diamond$-realization function on M. Then M is an L-model and $M, \sigma \models I$.*

Theorem 3.2 *The standard L-model of an L-normal model generator I is a least L-model of I.*

3.3 Fixpoint Semantics

We now return to the direct consequence operator $T_{L,P}$ for an L-MProlog program P. Given an L-normal model generator I, how can $T_{L,P}(I)$ be defined? Basing on the axioms of L, I is first extended to the "L-saturation of I" denoted by $Sat_L(I)$, which is a set of atoms. Next, "L-instances of program clauses" of P are "applied" to the atoms of $Sat_L(I)$. This is done by the operator $T_{0L,P}$. The set $T_{0L,P}(Sat_L(I))$ is a model generator but not necessary in L-normal form. Finally, the "normalization operator" NF_L converts $T_{0L,P}(Sat_L(I))$ to an L-normal model generator. $T_{L,P}(I)$ is defined as $NF_L(T_{0L,P}(Sat_L(I)))$.

We will define a pre-order $\preceq_L$ between modal operators for each specific logic L to decide whether a given modality is an L-instance of another one. We require that $\Diamond_i \preceq_L \langle E\rangle_i \preceq_L \Box_i$, $\Diamond_i \preceq_L \langle X\rangle_i \preceq_L \Box_i$, and if $\nabla \preceq_L \langle E\rangle_i$ and $\nabla \neq \langle E\rangle_i$ then $\nabla \preceq_L \langle X\rangle_i$. Note that the condition of seriality plays an essential role here. As an example, we have the following definition.

Definition 3.13 For $L \in \{KDI4_s5, KD4_s5_s, KD45_{(m)}, KDI45, KD4I_g5_a, sCFG\}$, define $\preceq_L$ to be the least reflexive and transitive relation between modal operators such that $\Diamond_i \preceq_L \langle E\rangle_i \preceq_L \Box_i$, $\Diamond_i \preceq_L \langle X\rangle_i \preceq_L \Box_i$, and additionally, $\Box_i \preceq_L \Box_j$ and $\Diamond_j \preceq_L \Diamond_i$ if

- $L \in \{KDI4_s, KDI4, KDI4_s5, KDI45\}$ and $i \leq j$, or
- $L = KD4I_g5_a$ and $g(i) \subseteq g(j)$, or
- $L = sCFG$ and $\Box_j\varphi \to \Box_i\varphi$ is L-valid (for every φ).

Definition 3.14 An atom $\nabla^{(1)} \ldots \nabla^{(n)}\alpha$ is called an *L-instance* of an atom $\nabla'^{(1)} \ldots \nabla'^{(n)}\alpha'$ if there exists a substitution θ such that $\alpha = \alpha'\theta$ and, for $1 \leq i \leq n$, $\nabla^{(i)} \preceq_L \nabla'^{(i)}\theta$ (treating $\nabla'^{(i)}$ as an expression). A modality $\triangle$ is called an *L-instance* of $\triangle'$ if $\triangle E$ is an L-instance of $\triangle' E$ for some ground classical atom E. In that case, we also say that $\triangle'$ is *equal to* or *more general in L than* $\triangle$ (hereby we define a *pre-order between modalities*).

For example, an atom $\Box_1\Diamond_2 E$ is a $KDI4_s5$-instance of $\Box_2\langle F\rangle_1 E$, and the modality $\Box_1\Diamond_2$ is a $KDI4_s5$-instance of $\Box_2\langle F\rangle_1$.

Definition 3.15 Let $\boxdot$ be a universal modality in L-normal form and $\boxdot'$ a modal context of an L-MProlog program clause. We say that $\boxdot$ is an *L-context instance* of $\boxdot'$ if $\boxdot'\varphi \to \boxdot\varphi$ is L-valid (for every φ).

Observe that if the problem of checking validity in the *propositional* version of L is decidable then the problem of checking whether $\boxdot$ is an L-context instance of $\boxdot'$ is also decidable. For the multimodal logics of belief specified in Section 2.2, these two problems are decidable and the latter is much simpler. Let $\boxdot$ be a universal modality in L-normal form and $\boxdot'$ a modal context of an L-MProlog program clause. For $L \in \{KDI4_s5, KD4_s5_s, KDI45, KD45_{(m)}\}$, it is easily seen that $\boxdot$ is an L-context instance of $\boxdot'$ iff $\boxdot = \boxdot'$ or one of the following condition holds:

- $L \in \{KDI4_s5, KD4_s5_s\}$ and $\boxdot$ is an L-instance of $\boxdot'$;
- $L = KDI45$, $\boxdot' = \Box_i$, $\boxdot$ is not empty, and every modal operator $\Box_j$ of $\boxdot$ satisfies $j \leq i$.

For $L = KD4I_g5_a$, the problem of checking L-context instance is decidable because the satisfiability problem in the propositional logic $KD4I_g5_a$ is decidable [12]. For $L = sCFG$, the problem of checking L-context instance is decidable by Corollary 2.2.

Definition 3.16 Let φ and φ' be program clauses with empty modal context, $\boxdot$ a universal modality in L-normal form, and $\boxdot'$ a modal context of an L-MProlog program clause. We say that $\boxdot\varphi$ is an *L-instance* of (a program clause) $\boxdot'\varphi'$ if $\boxdot$ is an L-context instance of $\boxdot'$ and there exists a substitution θ such that $\varphi = \varphi'\theta$.

For example, $\boxdot$ is a $KDI4_s5$-context instance of $\boxdot'$ iff $\boxdot$ is a $KDI4_s5$-instance of $\boxdot'$ (i.e. either $\boxdot$ and $\boxdot'$ are empty or $\boxdot = \Box_i$, $\boxdot' = \Box_j$, and $i \leq j$), and we have that $\Box_1(p(a) \leftarrow q(a))$ is a $KDI4_s5$-instance of $\Box_2(p(x) \leftarrow q(x))$.

We now give definitions concerning Sat_L, $T_{0L,P}$, and NF_L.

Definition 3.17 The *saturation operator Sat_L* is specified by a finite set of forward rules. Given an L-normal model generator I, $Sat_L(I)$ is the least extension of I that contains all ground atoms in almost L-normal labeled form that are derivable from some atom in I using the rules specifying Sat_L.

Definition 3.18 For $L \in \{KDI4_s5, KD4_s5_s, KD45_{(m)}, KDI45, KD4I_g5_a, sCFG\}$, Sat_L is specified by the rules specifying Ext_L plus the rules given below, in which formulas in both sides are required to be in almost L-normal labeled form if $L \notin \{KD4I_g5_a, sCFG\}$, while the restriction (*) of Definition 3.11 is discarded.[6]

[6]The operator Sat_L acts on sets of atoms, which may contain unlabeled existential modal operators, and it is thus not suitable for replacing Ext_L in the construction of the standard L-models of L-normal model generators.

$L = KDI4_s5$: $\Box_i E \to \Box_m \Box_i E$
$\langle F\rangle_i E \to \Box_m \Diamond_i E$
(Recall that m is the maximal modal index.)

$L = KD4_s5_s$: $\Box_i E \to \Box_j \Box_i E$
$\langle F\rangle_i E \to \Box_j \Diamond_i E$

$L = KD45_{(m)}$: $\triangle\Box_i E \to \triangle\Box_i \Box_i E$
$\triangle\langle F\rangle_i E \to \triangle\Box_i \Diamond_i E$

$L = KDI45$: $\triangle\Box_i E \to \triangle\Box_i \Box_i E$
$\triangle\nabla E \to \triangle\Box_i \Diamond_i E$ if $\Diamond_i \preceq_L \nabla$
$\triangle\Box_i \nabla_j E \to \triangle\Diamond_j E$ if $i > j$
$\triangle\langle F\rangle_i \nabla_j E \to \triangle\Diamond_i E$ if $i > j$

$L = KD4I_g5_a$: $\triangle\langle F\rangle_i E \to \triangle\Box_i \Diamond_i E$ if $g(i)$ is a singleton
$\triangle\nabla\nabla' E \to \triangle\Diamond_i E$ if $\Diamond_i \preceq_L \nabla$ and $\Diamond_i \preceq_L \nabla'$

$L = sCFG$: $\triangle\nabla^{(1)} \dots \nabla^{(k)}\alpha \to \triangle\Diamond_i \alpha$
if $\Box_i\varphi \to \Box_{j_1} \dots \Box_{j_k}\varphi$ is an axiom of L
and $\Diamond_{j_t} \preceq_L \nabla^{(t)}$ for all $1 \le t \le k$

As an example, for $L = KDI4_s5$, we have

$$Sat_L(\{\Box_2 p(a)\}) = \{\Box_2 p(a), \Box_1 p(a), \Box_m \Box_2 p(a), \Box_m \Box_1 p(a)\}.$$

When computing the least fixpoint of a modal logic program, whenever an atom of the form $\triangle\Diamond_i E$ is introduced, we "fix" the $\Diamond_i$ by replacing the atom by $\triangle\langle E\rangle_i E$. This leads to the following definition.

Definition 3.19 The *forward labeled 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 labeled form of $\Diamond_1 s(a)$ is $\langle s(a)\rangle_1 s(a)$.

Definition 3.20 Let P be an L-MProlog program. The *operator* $T_{0L,P}$ is defined as follows: for a set I of ground atoms in almost L-normal labeled form, $T_{0L,P}(I)$ is the least (w.r.t. $\subseteq$) model generator such that if $\boxdot(A \leftarrow B_1, \dots, B_n)$ is a ground L-instance of some program clause of P and $\triangle$ is a maximally general[7] ground modality in L-normal labeled form such that $\triangle$ is an L-instance of $\boxdot$ and $\triangle B_i$ is an L-instance of some atom of I (for every $1 \le i \le n$), then the forward labeled form of $\triangle A$ belongs to $T_{0L,P}(I)$.

For example, if P consists of the only clause $\Box_2(\Diamond_1 p(x) \leftarrow q(x), r(x), \Box_1 s(x), \Diamond_2 t(x))$ and $I = \{\langle q(a)\rangle_1 q(a), \langle q(a)\rangle_1 r(a), \Box_2\Box_2 s(a), \Box_2\langle t(a)\rangle_1 t(a)\}$ and $L = KDI4_s5$, then $T_{0L,P}(I) = \{\langle q(a)\rangle_1\langle p(a)\rangle_1 p(a)\}$.

Definition 3.21 The *normalization operator* NF_L is specified by a finite set of forward rules. Given a model generator I, $NF_L(I)$ is the set of all ground atoms in L-normal labeled form that are derivable from some atom of I using the rules specifying NF_L.

We require that if I is a singleton then $NF_L(I)$ is also a singleton. If there are no conditions on L-normal form of atoms, then the set of rules specifying NF_L is empty and $NF_L(I) = I$.

Definition 3.22 For $L \in \{KDI4_s5, KD4_s5_s, KD45_{(m)}, KDI45, KD4I_g5_a, sCFG\}$, NF_L is specified by the following rules, in which formulas in both sides are required to be in almost L-normal labeled form, and ∇_i is $\Box_i$ or $\langle E\rangle_i$.

[7]w.r.t. the pre-order between modalities described earlier for L

$L \in \{KDI4_s5, KD4_s5_s\}$:	$\nabla'_j \nabla_i E \to \nabla_i E$
$L = KD45_{(m)}$:	$\triangle\nabla'_i \nabla_i E \to \triangle\nabla_i E$
$L = KDI45$:	$\triangle\nabla'_j \nabla_i E \to \triangle\nabla_i E$ if $j \leq i$
$L = KD4I_g5_a$:	$\triangle\nabla_i \nabla'_i E \to \triangle\nabla'_i E$ if $g(i)$ is a singleton
$L = sCFG$:	no rules

As an example, for $L = KDI4_s5$, we have $NF_L(\{\langle q(a)\rangle_1 \langle p(a)\rangle_1 p(a)\}) = \{\langle p(a)\rangle_1 p(a)\}$.

Definition 3.23 Define $T_{L,P}(I) = NF_L(T_{0L,P}(Sat_L(I)))$.

We give here a digression about fixpoints. Let $T : \mathcal{P}(U) \to \mathcal{P}(U)$ be an operator that maps each subset of U to a subset of U. T is said to be *monotonic* if for every subsets V and V' of U, if $V \subseteq V'$ then $T(V) \subseteq T(V')$. T is said to be *continuous* if for every set $\mathcal{V}$ of subsets of U, $T(\bigcup \mathcal{V}) = \bigcup\{T(V) \mid V \in \mathcal{V}\}$. T is said to be *compact* if for every subset V of U and for every $a \in T(V)$, there exists a finite subset V' of V such that $a \in T(V')$. It is known that if T is monotonic and compact then T is continuous (see, e.g., [30]).

Lemma 3.3 *The operator $T_{L,P}$ is monotonic and continuous, and it has the least fixpoint $T_{L,P}\uparrow\omega = \bigcup_{n=0}^{\omega} T_{L,P}\uparrow n$, where $T_{L,P}\uparrow 0 = \emptyset$, and $T_{L,P}\uparrow n = T_{L,P}(T_{L,P}\uparrow(n-1))$ for $n > 0$.*

Proof. Clearly, all the operators Sat_L, $T_{0L,P}$ and NF_L are increasingly monotonic and compact. It follows that $T_{L,P}$ is monotonic, compact, and continuous. The second assertion of the lemma follows from the Kleen theorem. •

Notation 3.24 Denote the least fixpoint $T_{L,P}\uparrow\omega$ by $I_{L,P}$ and its standard L-model by $M_{L,P}$.

Definition 3.25 Let P be an L-MProlog program. An L-normal model generator I is called an *L-model generator of P* if $T_{L,P}(I) \subseteq I$.

As a property of the least fixpoint, $I_{L,P}$ is the least (w.r.t. $\subseteq$) L-model generator of P.

Example 3.2 Consider the following program P in $L = KDI4_s5$:

$$\begin{array}{ll} \Diamond_1 s(a) \leftarrow & \Box_1(q(x) \leftarrow r(x), s(x)) \\ \Box_1(\Box_1 r(x) \leftarrow s(x)) & \Box_2(p(x) \leftarrow \Diamond_2 q(x)) \end{array}$$

The least L-model generator of P is $I_{L,P} = \{\langle s(a)\rangle_1 s(a), \Box_1 r(a), \langle s(a)\rangle_1 q(a), \Box_2 p(a), \Box_1 p(a)\}$.

We give below expected results about the fixpoint semantics. The proofs of Lemmas 3.4 and 3.5 for $L \in \{KDI4_s5,\ KD4_s5_s,\ KD45_{(m)},\ KDI45,\ KD4I_g5_a,\ sCFG\}$ are given in [27].

Lemma 3.4 *For an L-MProlog program P, $P \vDash_L I_{L,P}$.*

Lemma 3.5 *Let P be an L-MProlog program and I an L-model generator of P. Then the standard L-model of I is an L-model of P.*

Theorem 3.6 *For an L-MProlog program P, $M_{L,P}$ is a least L-model of P.*

Proof. By Lemma 3.5, $M_{L,P}$ is an L-model of P. Let M be an arbitrary L-model of P. By Lemma 3.4, $M \vDash I_{L,P}$. Hence, by Theorem 3.2, $M_{L,P} \leq M$, and $M_{L,P}$ is a least L-model of P. •

3.4 SLD-Resolution

The fixpoint semantics can be viewed as a bottom-up method for computing answers. It repeatedly applies clauses of a given program P in order to compute the set $I_{L,P}$ of facts derivable in L from the program. Given an atom α from $I_{L,P}$, the process of tracing back the derivation of α in L from P is called top-down, because it reduces the atom, treated as a goal, to subgoals. A more general problem is to find answers for an L-MProlog goal w.r.t. an L-MProlog program. We study this problem using SLD-resolution.

The main work in developing an SLD-resolution calculus for L-MProlog is to specify a reverse analogue of the operator $T_{L,P}$. While $T_{L,P}$ acts on model generators (with only ground atoms), the expected reverse analogue of $T_{L,P}$ will act on goals (with variables). The operator $T_{L,P}$ is a composition of Sat_L, $T_{0L,P}$, and NF_L. So, we have to investigate reversion of these operators.

Definition 3.26 A *goal* is a clause of the form $\leftarrow \alpha_1, \ldots, \alpha_k$, where each α_i is an atom.

The following definition concerns reversion of the operator $T_{0L,P}$.

Definition 3.27 Let $G = \leftarrow \alpha_1, \ldots, \alpha_i, \ldots, \alpha_k$ be a goal, $\alpha_i = \triangle' A'$ be the "selected atom" with $\triangle'$ in L-normal labeled form and A' called the "selected head atom", and $\varphi = \boxdot(A \leftarrow B_1, \ldots, B_n)$ be a program clause. Then G' is *derived* from G and φ in L using an mgu θ, and called an *L-resolvent* of G and φ, if the following conditions hold:

- $\triangle'$ is an L-instance of a universal modality $\boxdot'$ and $\boxdot'(A \leftarrow B_1, \ldots, B_n)$ is an L-instance of the program clause φ;
- θ is an mgu of A' and the forward labeled form of A;
- 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 unique $KDI4_s5$-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 unique $KDI4_s5$-resolvent of $\leftarrow \langle Y \rangle_1 \Box_1 r(x), \langle X \rangle_1 s(x)$ 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 $\triangle' = \langle Y \rangle_1$).

As a reverse analogue of the operator Sat_L, we provide the operator $rSat_L$.

Definition 3.28 The *operator $rSat_L$* is specified by a finite set of backward rules. We say that $\beta = rSat_L(\alpha)$ *using an $rSat_L$ rule* $\alpha' \leftarrow \beta'$ if $\alpha \leftarrow \beta$ is of the form $\alpha' \leftarrow \beta'$. We write $\beta = rSat_L(\alpha)$ to denote that "$\beta = rSat_L(\alpha)$ using some $rSat_L$ rule".

We require that one of the $rSat_L$ rules is the *backward labeling rule* $\triangle \Diamond_i E \leftarrow \triangle \langle X \rangle_i E$ with X being a fresh[8] atom variable. We call $\triangle \langle X \rangle_i E$ a *backward labeled form* of $\triangle \Diamond_i E$.

Definition 3.29 For $L \in \{KDI4_s5, KD4_s5_s, KD45_{(m)}, KDI45, KD4I_g5_a, sCFG\}$, $rSat_L$ is specified by the following rules, in which formulas in both sides are required to be in almost L-normal labeled form if $L \notin \{KD4I_g5_a, sCFG\}$:

[8]This means that *standardizing* is also needed for atom variables.

$$\begin{array}{ll}
\text{a common rule} & \\
\text{for } L \neq sCFG: & \triangle\Diamond_i E \leftarrow \triangle\langle X\rangle_i E \;\text{ for } X \text{ being a fresh atom variable} \\
 & \\
L = KDI4_s5: & \triangle\nabla_i\alpha \leftarrow \triangle\Box_j\alpha \;\text{ if } i \leq j \\
 & \triangle\Diamond_i E \leftarrow \triangle\Diamond_j E \;\text{ if } i > j \\
 & \nabla\nabla' E \leftarrow \nabla' E \;\text{ if } \nabla' \text{ is of the form } \Box_i \text{ or } \Diamond_i \\
 & \\
L = KD4_s5_s: & \triangle\nabla_i\alpha \leftarrow \triangle\Box_i\alpha \\
 & \nabla\nabla' E \leftarrow \nabla' E \;\text{ if } \nabla' \text{ is of the form } \Box_i \text{ or } \Diamond_i \\
 & \\
L = KD45_{(m)}: & \triangle\nabla_i\alpha \leftarrow \triangle\Box_i\alpha \\
 & \triangle\nabla_i\nabla'_i E \leftarrow \triangle\nabla'_i E \;\text{ if } \nabla'_i \text{ is of the form } \Box_i \text{ or } \Diamond_i \\
 & \\
L = KDI45: & \triangle\nabla_i\alpha \leftarrow \triangle\Box_j\alpha \;\text{ if } i \leq j \\
 & \triangle\Diamond_i E \leftarrow \triangle\Diamond_j E \;\text{ if } i > j \\
 & \triangle\Box_i\Box_j\alpha \leftarrow \triangle\Box_i\alpha \;\text{ if } i \geq j \\
 & \triangle\Box_i\Diamond_i E \leftarrow \triangle\Diamond_i E \\
 & \triangle\Box_i\alpha \leftarrow \triangle\Box_j\Box_i\alpha \;\text{ if } i < j \\
 & \triangle\Diamond_i E \leftarrow \triangle\langle X\rangle_i\Diamond_i E \;\text{ for } X \text{ being a fresh atom variable} \\
 & \\
L = KD4I_g5_a: & \triangle\Diamond_i E \leftarrow \triangle\Diamond_j E \;\text{ if } g(i) \supset g(j) \\
 & \triangle\nabla_i\alpha \leftarrow \triangle\Box_j\alpha \;\text{ if } g(i) \subseteq g(j) \\
 & \triangle\Box_i\Box_i\alpha \leftarrow \triangle\Box_i\alpha \\
 & \triangle\Box_i\alpha \leftarrow \triangle\langle X\rangle_i\Box_i\alpha \;\text{ if } g(i) \text{ is a singleton and} \\
 & \qquad X \text{ is a fresh atom variable} \\
 & \triangle\nabla_i\Diamond_i E \leftarrow \triangle\Diamond_i E \;\text{ if } g(i) \text{ is a singleton} \\
 & \triangle\Diamond_i E \leftarrow \triangle\langle X\rangle_j\Diamond_i E \;\text{ if } g(i) \supseteq g(j) \text{ and} \\
 & \qquad X \text{ is a fresh atom variable} \\
 & \\
L = sCFG: & \triangle\Diamond_i\alpha \leftarrow \triangle\langle X\rangle_i\alpha \;\text{ for } X \text{ being a fresh atom variable} \\
 & \triangle\nabla^{(1)}\ldots\nabla^{(k)}\alpha \leftarrow \triangle\Box_i\alpha \;\text{ if} \\
 & \qquad \Box_i\varphi \to \Box_{j_1}\ldots\Box_{j_k}\varphi \in Axioms(L) \text{ and} \\
 & \qquad \nabla^{(t)} \preceq_L \Box_{j_t} \text{ for all } 1 \leq t \leq k \\
 & \triangle\Diamond_i\alpha \leftarrow \triangle\Diamond_{j_1}\ldots\Diamond_{j_k}\alpha \;\text{ if} \\
 & \qquad \Box_i\varphi \to \Box_{j_1}\ldots\Box_{j_k}\varphi \in Axioms(L) \\
 & \triangle\nabla\alpha \leftarrow \triangle\Box_i\alpha \;\text{ if } \nabla \preceq_L \Box_i
\end{array}$$

Definition 3.30 Let $G = \leftarrow \alpha_1, \ldots, \alpha_i, \ldots, \alpha_k$ be a goal. If $\alpha'_i = rSat_L(\alpha_i)$ using an $rSat_L$ rule φ, then $G' = \leftarrow \alpha_1, \ldots, \alpha_{i-1}, \alpha'_i, \alpha_{i+1}, \ldots, \alpha_k$ is *derived* from G and φ, and we call G' an *(L-)resolvent* of G and φ, and α_i the *selected atom* of G.

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 $\Diamond_2$.

As a reverse analogue of the operator NF_L, we provide the operator rNF_L.

Definition 3.31 The *operator* rNF_L is specified by a finite set of backward rules. We say that $\beta =_\theta rNF_L(\alpha)$ *using an* rNF_L *rule* $\alpha' \leftarrow \beta'$ if θ is an mgu such that $\alpha\theta \leftarrow \beta$ is of the form $\alpha' \leftarrow \beta'$. We write $\beta =_\theta rNF_L(\alpha)$ if "$\beta =_\theta rNF_L(\alpha)$ using some rNF_L rule".

Definition 3.32 For $L \in \{KDI4_s5, KD4_s5_s, KD45_{(m)}, KDI45, KD4I_g5_a, sCFG\}$, rNF_L is specified by the following rules, in which formulas in both sides are required to be in almost L-normal labeled form, ∇_i is $\Box_i$ or $\langle E\rangle_i$, and X is a fresh atom variable.

$L \in \{KDI4_s5, KD4_s5_s\}$:	$\nabla_i E \leftarrow \langle X\rangle_j \nabla_i E$
$L = KD45_{(m)}$:	$\triangle\nabla_i E \leftarrow \triangle\langle X\rangle_i \nabla_i E$
$L = KDI45$:	$\triangle\nabla_i E \leftarrow \triangle\langle X\rangle_j \nabla_i E$ if $j \leq i$
$L = KD4I_g5_a$:	$\triangle\nabla_i E \leftarrow \triangle\langle X\rangle_i \nabla_i E$ if $g(i)$ is a singleton
$L = sCFG$:	no rules

As an example, for $L = KDI4_s5$, we have $\langle Y\rangle_1\langle E\rangle_2 E =_\theta rNF_L(\langle X\rangle_2 E)$ with $\theta = \{X/E\}$ and Y being a fresh atom variable.

Definition 3.33 Let $G = \leftarrow \alpha_1, \ldots, \alpha_i, \ldots, \alpha_k$ be a goal. If $\alpha_i' =_\theta rNF_L(\alpha_i)$ using an rNF_L rule φ, then $G' = \leftarrow \alpha_1\theta, \ldots, \alpha_{i-1}\theta, \alpha_i', \alpha_{i+1}\theta, \ldots, \alpha_k\theta$ is *derived* from G and φ using the mgu θ, and we call G' an *(L-)resolvent* of G and φ, and α_i the *selected atom* of G.

Observe that $rSat_L$ rules and rNF_L rules are similar to program clauses and the way of applying them is similar to the way of applying classical program clauses, except that we do not need mgu's for $rSat_L$ rules.

We now define SLD-derivation and SLD-refutation. We say that an expression φ is a *variant* of an expression ψ if there exist substitutions θ and γ such that $\varphi = \psi\theta$ and $\psi = \varphi\gamma$.

Definition 3.34 Let P be an L-MProlog program and G be a goal. An *SLD-derivation* from $P \cup \{G\}$ in L consists of a (finite or infinite) sequence $G_0 = G, G_1, \ldots$ of goals, a sequence $\varphi_1, \varphi_2, \ldots$ of variants of program clauses of P, $rSat_L$ rules, or rNF_L rules, and a sequence $\theta_1, \theta_2, \ldots$ of mgu's such that if φ_i is a variant of a program clause or an rNF_L rule then G_i is derived from G_{i-1} and φ_i in L using θ_i, else $\theta_i = \varepsilon$ (the empty substitution) and G_i is derived from G_{i-1} and (the $rSat_L$ rule variant) φ_i.

We require that each φ_i in the above definition does not have any variable or atom variable which already appears in the derivation up to G_{i-1}. This can be achieved by subscripting variables and atom variables in G by 0 and in φ_i by i. This process of renaming variables is usually called *standardizing* the variables *apart* (see [18]). Each φ_i is called an *input clause/rule* of the derivation.

Definition 3.35 An *SLD-refutation* of $P \cup \{G\}$ in L is a finite SLD-derivation from $P \cup \{G\}$ in L which has the empty clause (denoted by $\diamond$) as the last goal in the derivation.

Definition 3.36 Let P be an L-MProlog program and G be a goal. A *computed answer* θ in L of $P \cup \{G\}$ is the substitution obtained by restricting the composition $\theta_1 \ldots \theta_n$ to the variables and atom variables of G, where $\theta_1, \ldots, \theta_n$ is the sequence of mgu's used in an SLD-refutation of $P \cup \{G\}$ in L.

Example 3.3 Consider the following program P and the goal $G = \leftarrow \Box_1 p(x)$ in $L = KDI4_s5$:

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

Here is an SLD-refutation of $P \cup \{G\}$ in L with computed answer $\{x/a\}$:

Goals	Input clauses/rules	MGUs
$\leftarrow \Box_1 p(x)$		
$\leftarrow \Box_1 \Diamond_2 q(x)$	φ_1	$\{x_1/x\}$
$\leftarrow \Diamond_2 q(x)$	$rSat_L : \Box_1\Diamond_2 E \leftarrow \Diamond_2 E$	
$\leftarrow \Diamond_1 q(x)$	$rSat_L : \triangle\Diamond_2 E \leftarrow \triangle\Diamond_1 E$	
$\leftarrow \langle X\rangle_1 q(x)$	$rSat_L : \triangle\Diamond_1 E \leftarrow \triangle\langle X\rangle_1 E$	
$\leftarrow \langle X\rangle_1 r(x), \langle X\rangle_1 s(x)$	φ_2	$\{x_5/x\}$
$\leftarrow \Box_1 r(x), \langle X\rangle_1 s(x)$	$rSat_L : \triangle\nabla_1 E \leftarrow \triangle\Box_1 E$	
$\leftarrow \langle Y\rangle_1\Box_1 r(x), \langle X\rangle_1 s(x)$	$rNF_L : \Box_1 E \leftarrow \langle Y\rangle_1\Box_1 E$	
$\leftarrow \langle Y\rangle_1 s(x), \langle X\rangle_1 s(x)$	φ_3	$\{x_8/x\}$
$\leftarrow \langle X\rangle_1 s(a)$	φ_4	$\{x/a, Y/s(a)\}$
$\diamond$	φ_4	$\{X/s(a)\}$

We give below an expected theorem on soundness and completeness of SLD-resolution for L-MProlog. Its proof for $L \in \{KDI4_s5,\ KD4_s5_s,\ KD45_{(m)},\ KDI45,\ KD4I_g5_a,\ sCFG\}$ is given in [27].

Theorem 3.7 (Soundness and Completeness of SLD-Resolution) *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 $P \cup \{G\}$. Conversely, for every correct answer θ in L for $P \cup \{G\}$, there exists a computed answer γ in L for $P \cup \{G\}$ such that $G\theta = G\gamma\delta$ for some substitution δ.*

4 MDatalog and Modal Deductive Databases

In this section, we give definitions for modal deductive databases and define a query language called MDatalog for such databases. Let L be one of the modal logics considered in this paper. We divide the group of such logics into two subgroups $\mathcal{BMD}$ (bounded modal depth) and $\mathcal{UMD}$ (unbounded modal depth) as follows: $\mathcal{BMD} = \{KDI4_s5, KDI45, KD4_s5_s, KD45_{(m)}\}$ and $\mathcal{UMD} = \{sCFG, KD4I_g5_a\}$.[9]

A (database) *schema* is a finite set of predicates, where each predicate p has a fixed arity, denoted by $arity(p)$, and is either *extensional* (*edb*) or *intensional* (*idb*). We denote the set of *edb* (resp. *idb*) predicates of a schema S by $edb(S)$ (resp. $idb(S)$). Informally, in a deductive database, extensional predicates are explicitly specified by relations, while intensional predicates are defined by a logic program that may use the extensional predicates.

Definition 4.1 An MProlog program clause without function symbols is *range-restricted* (or *allowed*) if every variable occurring in the head also occurs in the body. An MProlog program over a schema S is an MProlog program whose predicates belong to S and whose predicates that occur in heads of program clauses belong to $idb(S)$. An *L-MDatalog program* over a schema S is an L-MProlog program over S which is free from function symbols and contains only range-restricted clauses.

Definition 4.2 An n-ary *L-tuple* is an ordered pair $(\triangle, \mathfrak{t})$, where $\mathfrak{t}$ is a classical n-ary tuple of constant symbols and $\triangle$ is a ground modality in almost L-normal labeled form. An n-ary *L-relation* is a set of n-ary L-tuples. An L-relation is an n-ary L-relation for some n. An L-relation is in *L-normal labeled form* if its tuples are of the form $(\triangle, \mathfrak{t})$ with $\triangle$ in L-normal labeled form.

When it is clear from the context, we also write "tuple" to mean "L-tuple".

When an L-relation is attached to (or named by) a predicate p of the same arity, we call the relation an L-relation of p. If $(\triangle, \mathfrak{t})$ is a tuple in an L-relation of a predicate p then we also treat

[9]In [27], we consider larger groups of modal logics:

- $\mathcal{BMD} = \{KDI4_s5, KDI45, KD4_s5_s, KD45_{(m)}, KD5, KD45, S5, KD, T, KDB, B\}$,
- $\mathcal{UMD} = \{sCFG, KDI4_s, KDI4, KD4I_g5_a, KD4, S4\}$.

it as the atom $\triangle p(\mathbf{t})$. Conversely, an ground atom $\triangle p(\mathbf{t})$ in almost L-normal labeled form can be treated as an L-tuple related with the predicate p. In this way, an L-relation of a predicate p can be treated as a set of atoms of p, and conversely, a set of ground atoms of p which are in almost L-normal labeled form can be treated as an L-relation of p.

A *database instance* I over a schema S in L is a mapping that maps each predicate $p \in S$ to an L-relation of arity $arity(p)$ in L-normal labeled form. A database instance over S can be treated as a set of atoms of the predicates of S. Conversely, a set I of ground atoms of the predicates of S which are in L-normal labeled form can be treated as a database instance over S in L.

An *extensional database (edb) instance* I over $edb(S)$ in L is a database instance over $edb(S)$ in L such that there exists an L-MDatalog program P_I consisting of unary clauses with the property that $I = I_{L,P_I}$, i.e. $I = T_{L,P_I} \uparrow \omega$. We call P_I the *source program* of I. An *edb* instance I can be either explicitly given or specified by P_I.[10] In the first case, we require only the existence of P_I and will not use it for computation. In the second case, it can be shown that

$$I = Ext_L(NF_L(\{\boxdot E \mid \boxdot E \in P_I\} \cup \{\boxdot\langle E\rangle_i E \mid \boxdot\Diamond_i E \in P_I\})).$$

Definition 4.3 A *modal deductive database* over a schema S in L consists of an *edb* instance I over $edb(S)$ in L and an L-MDatalog program P over S. The program P can be treated as the function P_L that maps I to a database instance over $idb(S)$ such that $P_L(I)$ is the least (w.r.t. $\subseteq$) L-model generator J such that $T_{L,P}(I \cup J) = J$.

The program P_{ddb} given in Example 2.1 is an L-MDatalog program. All the predicates used in that program are intensional. To treat unary clauses (e.g. $\Box_1 likes(Jan, cola) \leftarrow$) as extensional data, we can change predicate *likes* in those clauses to an extensional predicate *likes_* and add to the program the clauses $\Box_i(likes(x,y) \leftarrow likes\_(x,y))$, for $i = 1,2,3$.

Let $T_{L,P,I}$ be the operator defined by $T_{L,P,I}(J) = T_{L,P}(I \cup J)$. Then $T_{L,P,I}$ is monotonic and continuous, and $P_L(I)$ is the least fixpoint of $T_{L,P,I}$ specified by $T_{L,P,I} \uparrow \omega = \bigcup_{0 \leq k \leq \omega} T_{L,P,I} \uparrow k$, where $T_{L,P,I} \uparrow k$ is defined in a similar way as $T_{L,P} \uparrow k$. By the following lemma, this definition of $P_L(I)$ is compatible with the semantics of MProlog programs:

Lemma 4.1 *Let P be an L-MDatalog program over a schema S and I be an edb instance over $edb(S)$ in $L \in \mathcal{BMD} \cup \mathcal{UMD}$. Then $P_L(I) \cup I = T_{L,(P \cup P_I)} \uparrow \omega$, where P_I is the source program of I. As a consequence, the standard L-model of $P_L(I) \cup I$ is a least L-model of $P \cup P_I$.*

Proof. Note that I is the least fixpoint of T_{L,P_I}, and the predicates occurring in P_I, i.e. $edb(S)$, are not defined by P. It is easy to see that $P_L(I) \cup I$ is contained in the least fixpoint of $T_{L,(P \cup P_I)}$, and it is also a fixpoint of $T_{L,(P \cup P_I)}$, hence it is the least fixpoint of $T_{L,(P \cup P_I)}$. The second assertion follows from Theorem 3.6. •

Definition 4.4 An *L-MDatalog query* over a schema S is a pair $(P, \varphi(x_1, \ldots, x_k))$, where P is an L-MDatalog program over S and $\varphi(x_1, \ldots, x_k)$ is a positive formula without quantifiers over (the signature) S such that $x_1, \ldots, x_k$ are all the variables of φ and if $\psi_1 \vee \psi_2$ is a subformula of φ then ψ_1 and ψ_2 have the same variables. An L-MDatalog query (P, φ) over S takes as input an *edb* instance I over $edb(S)$ in L and returns the classical relation of tuples $(c_1, \ldots, c_k)$ of constant symbols such that $P \cup P_I \vDash_L \varphi(c_1, \ldots, c_k)$.

Proposition 4.2 *Every L-MDatalog query $(P, \varphi(x_1, \ldots, x_k))$ over a schema S, where $L \in \mathcal{BMD} \cup \mathcal{UMD}$, can be transformed in polynomial time to an L-MDatalog query $(P', q(x_1, \ldots, x_k))$ over $S \cup \{q\}$, where q is an idb predicate, such that for every edb instance I over $edb(S)$ in L and every constant symbols $c_1, \ldots, c_k$,*

$$P \cup P_I \vDash_L \varphi(c_1, \ldots, c_k) \;\; \textit{iff} \;\; P' \cup P_I \vDash_L q(c_1, \ldots, c_k).$$

This proposition can be proved in a similar way as done in [27] for the proposition on the expressiveness of MProlog.

[10]Sometimes, when $L \in \mathcal{UMD}$, I may be infinite and cannot be explicitly given.

5 Data Complexity

Data complexity is measured when the query is fixed and the extensional database is taken as input. In this section, we show that for $L \in \mathcal{BMD}$ the data complexity of L-MDatalog is in PTIME, i.e. for every L-MDatalog query $(P, \varphi(x_1, \ldots, x_k))$ over a schema S and every constant symbols $c_1, \ldots, c_k$, the problem of checking $P \cup P_I \vDash_L \varphi(c_1, \ldots, c_k)$ for an input *edb* instance I over *edb*(S) in L is in PTIME. Here, I but not P_I is taken as input, but the case when P_I is taken as input does not differ much, because I can be computed from P_I in polynomial time when $L \in \mathcal{BMD}$. Since the data complexity of Datalog is complete in PTIME (see, e.g., [16]), the data complexity of L-MDatalog for $L \in \mathcal{BMD}$ is also complete in PTIME.

Theorem 5.1 *The data complexity of L-MDatalog for $L \in \mathcal{BMD}$ is in PTIME.*

Proof. Let $(P, \varphi(x_1, \ldots, x_k))$ be an L-MDatalog query over a schema S and I be an input *edb* instance for this query. Recall that the size of a formula set is the sum of the lengths of its formulas. Let d be the size of P and n be the size of I. Note that, for $\alpha \in T_{L,P,I} \uparrow h$ for some h, the modal depth of α is bounded by 1 for $L \in \{KDI4_s5, KD4_s5_s\}$, by m for $L = KDI45$, and by the modal depth of P for $L = KD45_{(m)}$. Denote this bound by e. Since P is fixed, both d and e are constants.

The key of this proof is that modal depths of atoms appearing in $T_{L,P,I} \uparrow \omega$ are bounded by e. Also observe that for any atom α, the sets $Sat_L(\{\alpha\})$ and $NF_L(\{\alpha\})$ can be computed in a finitely bounded number of steps. Fix some $h \geq 1$ and let $J = T_{L,P,I} \uparrow h$ and $\alpha \in J$.

The number of classical atoms that may occur in (the atoms of) J is of rank $O(n^d)$. Hence the size of J is of rank $O(n^{d(e+1)})$. It follows that the size of $Sat_L(I \cup J)$ and the number of steps needed for computing $Sat_L(I \cup J)$ from I and J are also of rank $O(n^{d(e+1)})$. The number of steps needed for computing $T_{0L,P}(Sat_L(I \cup J))$ from $Sat_L(I \cup J)$ is of rank $O(n^{d.d.(e+1)})$. The size of $T_{0L,P}(Sat_L(I \cup J))$ can be estimated in a similar way as the size of J and is of rank $O(n^{d(e+2)})$. The number of steps needed for computing $T_{L,P,I} \uparrow (h+1)$ from $T_{0L,P}(Sat_L(I \cup J))$ is of the same rank as the size of $T_{0L,P}(Sat_L(I \cup J))$. Therefore the number of steps needed to compute $T_{L,P,I} \uparrow (h+1)$ from $T_{L,P,I} \uparrow h$ is bounded by a polynomial of n. The size of $T_{L,P,I} \uparrow \omega$ can be estimated in the same way as the size of J and is of rank $O(n^{d(e+1)})$. Hence the number of steps needed to compute $T_{L,P,I} \uparrow \omega$ is bounded by a polynomial of n.

Consider the standard L-model M of $P_L(I) \cup I$. Since the possible worlds of M are identified by ground modalities with length bounded by e, the number of possible worlds of M is of rank $O(n^{d.e})$. The content of each possible world of M is the set of ground classical atoms that hold at the world, and hence has size of rank $O(n^d)$. The size of M, defined as the sum of the number of possible worlds, the sizes of the accessibility relations, and the sizes of the contents of the possible worlds, is thus of rank $O(n^{2.d.e})$. The size of $Ext_L(P_L(I) \cup I) \cup Serial_L$ and the number of steps needed for computing that set can be estimated in the same way as for $Sat_L(I \cup J)$ and are of rank $O(n^{d.e})$. Hence the number of steps needed to construct M is bounded by a polynomial of n.

By Lemma 4.1, M is a least L-model of $P \cup P_I$. Hence, for every constant symbols $c_1, \ldots, c_k$, $P \cup P_I \vDash_L \varphi(c_1, \ldots, c_k)$ iff $M \vDash \varphi(c_1, \ldots, c_k)$. Checking $M \vDash \varphi(c_1, \ldots, c_k)$ is done in polynomial time in the size of M and φ. Hence the data complexity of L-MDatalog is in PTIME. •

For $L \in \mathcal{UMD}$, data complexity of L-MDatalog is defined using P_I as input and under the assumption that the extensional database I is specified by P_I. (The problem is that I may be infinite even when P_I is finite.) We do not have an exact data complexity of L-MDatalog for the case $L \in \{KD4I_g5_a, sCFG\}$. As the data complexity of $KD4$-MDatalog is complete in PSPACE [24] and $KD4$ is a fragment of $KD4I_g5_a$ and a logic of $sCFG$, the data complexity of $KD4I_g5_a$-MDatalog and $sCFG$-MDatalog is PSPACE-hard.

6 Modal Relational Algebras

Let L be one of the modal logics considered in this paper. In this section, we define a modal relational algebra in L, called the L-SPCU algebra. This algebra extends the classical SPCU algebra (see, e.g., [1]) with some operators involving with modalities. We also compare L-SPCU algebra queries with *nonrecursive* L-MDatalog programs (defined in Definition 6.2).

In this section, we use I, J to denote L-relations.

The *L-SPCU algebra* is formed by the following operators:

Selection The two primitive forms are $\sigma_{j=c}$ and $\sigma_{j=k}$, where j, k are positive integers and c is a constant symbol. The operator $\sigma_{j=c}$ takes as input any L-relation I with arity $\geq j$ and returns as output an L-relation of the same arity. In particular, $\sigma_{j=c}(I) = \{(\triangle, \mathfrak{t}) \mid (\triangle, \mathfrak{t}) \in I \text{ and } \mathfrak{t}(j) = c\}$. The operator $\sigma_{j=k}$ is defined analogously for inputs with arity $\geq max\{j, k\}$: $\sigma_{j=k}(I) = \{(\triangle, \mathfrak{t}) \mid (\triangle, \mathfrak{t}) \in I \text{ and } \mathfrak{t}(j) = \mathfrak{t}(k)\}$. A composition of selections is written in the form $\sigma_{\varphi_1 \wedge \ldots \wedge \varphi_k}$ and defined by $\sigma_{\varphi_1 \wedge \ldots \wedge \varphi_k}(I) = \sigma_{\varphi_1}(\ldots(\sigma_{\varphi_k}(I))\ldots)$.

Projection The general form of this operator is $\pi_{j_1,\ldots,j_n}$, where $j_1, \ldots, j_n$ is a sequence of positive integers, possibly with repeats. This operator takes as input any L-relation with arity $\geq max\{j_1, \ldots, j_n\}$ and returns an L-relation with arity n. In particular, $\pi_{j_1,\ldots,j_n}(I) = \{(\triangle, (c_1, \ldots, c_n)) \mid (\triangle, \mathfrak{t}) \in I \text{ for some } \mathfrak{t} \text{ with } \mathfrak{t}(j_i) = c_i \text{ for } 1 \leq i \leq n\}$.

Cross-product This operator, denoted by $\times$, takes as input a pair of L-relations in L-normal labeled form with arbitrary arities k and h and returns an L-relation with arity $k + h$, which is also in L-normal labeled form. In particular, if $arity(I) = k$ and $arity(J) = h$, then $I \times J = \{(\triangle, (\mathfrak{t}(1), \ldots, \mathfrak{t}(k), \mathfrak{s}(1), \ldots, \mathfrak{s}(h))) \mid$ there exist $\triangle'$ and $\triangle''$ such that $(\triangle', \mathfrak{t}) \in I$, $(\triangle'', \mathfrak{s}) \in J$, and $\triangle$ is a maximal L-instance in L-normal labeled form of $\triangle'$ and $\triangle''\}$.

Union This operator, denoted by $\cup$, takes as input a pair of L-relations with the same arity and returns an L-relation with the same arity that is the sum of the input relations.

Context-shrink The two primitive forms are $\Box_i^-$ and $\Diamond_i^-$, where $1 \leq i \leq m$.[11] These operators take as input any L-relation I and return as output an L-relation in L-normal labeled form of the same arity. In particular, $\Box_i^-(I) = \{(\triangle, \mathfrak{t}) \mid \text{there exists } (\triangle\nabla, \mathfrak{t}) \in I \text{ such that } \Box_i \preceq_L \nabla\}$. The operator $\Diamond_i^-$ is defined analogously.

Context-stretch The two primitive forms are $\Box_i^+$ and $\Diamond_i^+$, where $1 \leq i \leq m$.[12] These operators take as input any L-relation I in L-normal labeled form and return as output an L-relation of the same arity. In particular, $\Box_i^+(I) = \{(\triangle\Box_i, \mathfrak{t}) \mid (\triangle, \mathfrak{t}) \in I\}$ and $\Diamond_i^+(I) = \{(\triangle\Diamond_i, \mathfrak{t}) \mid (\triangle, \mathfrak{t}) \in I\}$.

Context-selection The general form of this operator is $\sigma_{\boxdot}$, where $\boxdot$ is the modal context of an L-MDatalog program clause. This operator takes as input any L-relation I in L-normal labeled form and returns as output an L-relation in L-normal labeled form of the same arity. In particular, $\sigma_{\boxdot}(I) = \{(\triangle, \mathfrak{t}) \mid (\triangle, \mathfrak{t}) \in I$ and the universal modality $\boxdot'$ being a $\Box$-lifting form of $\triangle$ is an L-context instance of $\boxdot\}$.

Saturation This operator, denoted by Sat_L, takes as input any L-relation I in L-normal labeled form and returns as output an L-relation of the same arity. In particular, $Sat_L(I) = \{(\triangle, \mathfrak{t}) \mid$ there exists $\triangle'$ such that $(\triangle', \mathfrak{t}) \in I$ and $\triangle E \in Sat_L(\{\triangle' E\})$ for some $E\}$, where the latter operator Sat_L acts on model generators as defined in Section 3.3.

Labeling The general form of this operator is $Label_p$, where p is an n-ary predicate symbol. This operator takes as input any L-relation I with arity n and returns as output an L-relation of the same arity. In particular, $Label_p(I) = \{(\triangle, \mathfrak{t}) \mid (\triangle, \mathfrak{t}) \in I \text{ and } \triangle \text{ is not of the form } \triangle'\Diamond_i\} \cup \{(\triangle\langle p(c_1, \ldots, c_n)\rangle_i, (c_1, \ldots, c_n)) \mid (\triangle\Diamond_i, (c_1, \ldots, c_n)) \in I\}$.

Normalization This operator, denoted by NF_L, takes as input any L-relation I and returns as output an L-relation in L-normal labeled form and of the same arity. In particular, $NF_L(I) = \{(\triangle, \mathfrak{t}) \mid$ there exists $\triangle'$ such that $(\triangle', \mathfrak{t}) \in I$ and $\triangle E \in NF_L(\{\triangle' E\})$ for some $E\}$, where the latter operator NF_L acts on model generators as defined in Section 3.3.

[11] In [23], $\Box_i^-$ and $\Diamond_i^-$ are denoted by $\Box_i$ and $\Diamond_i$, respectively.

[12] In [23], $\Box_i^+$ and $\Diamond_i^+$ are denoted by $\Box_i^{\leftarrow}$ and $\Diamond_i^{\leftarrow}$, respectively.

Note that the operators $\times$, $\Box_i^-$, $\Diamond_i^-$, and $\sigma_{\boxdot}$ are dependent on the base logic L. However, for simplicity we do not attach the index L to these operators.

For $L \in \mathcal{BMD} \cup \mathcal{UMD}$, all the above operations except Sat_L can be effectively computed and return a finite L-relation if the input consists of finite L-relations. This is clear for selection, projection, union, context-shrink, context-stretch, context-selection, and labeling. The assertion holds for cross-product because, given two ground modal operators ∇ and ∇', there is at most one maximal modal operator ∇'' such that $\nabla'' \preceq_L \nabla$ and $\nabla'' \preceq_L \nabla'$. The assertion holds for normalization because the set $NF_L(\{\triangle' E\})$ is finite and can be effectively computed for every ground atom $\triangle' E$. Similarly, for $L \in \mathcal{BMD}$, the operation Sat_L can also be effectively computed and returns a finite L-relation if the input consists of finite L-relations. For $L \in \mathcal{UMD}$, the operator Sat_L may return an infinite L-relation even when the input consists of finite L-relations.

Definition 6.1 *L-SPCU (algebra) queries* are built from input L-relations and unary constant relations $I_L^c = \{(\boxdot, (c)) \mid \boxdot$ is a universal modality in L-normal labeled form$\}$, where c is a constant symbol, using the L-SPCU algebra operators.

Definition 6.2 A predicate p *directly depends* on a predicate q in an L-MDatalog program P if there exists a program clause φ of P containing p in the head and q in the body. Define the relation "*depends*" to be the transitive closure of the relation "directly depends". An L-MDatalog program P is *nonrecursive* if none of its predicates depends on itself.

Proposition 6.1 *Every L-MDatalog query $(P, \varphi(x_1, \ldots, x_k))$, where $L \in \mathcal{BMD} \cup \mathcal{UMD}$ and P is a nonrecursive L-MDatalog program, is equivalent to an L-SPCU query.*

Sketch. By Proposition 4.2, we can assume that $\varphi(x_1, \ldots, x_k)$ is of the form $q(x_1, \ldots, x_k)$, where q is a predicate. Since the L-SPCU algebra contains the union operator, it is sufficient to show that every L-relation *ans* defined by a nonrecursive L-MDatalog program clause is equivalent to an L-SPCU query. For simplicity, we show this using the following representative example

$$\boxdot(\Diamond_i ans(x, x, z, a) \leftarrow \Box_j R(x, b), \Diamond_k S(x, y), T(z)).$$

Let

$$\begin{aligned} Q_1 &= \sigma_{\boxdot}(\Box_j^-(Sat_L(\sigma_{2=b}(R)))), \\ Q_2 &= \pi_1(\sigma_{1=3}(Q_1 \times \Diamond_k^-(Sat_L(S)))), \\ Q_3 &= Q_2 \times Sat_L(T). \end{aligned}$$

Then *ans* is equivalent to

$$NF_L(Label_{ans}(\Diamond_i^+(\pi_{1,1,2,3}(Q_3 \times I_L^a)))).$$

●

The conversion of the above proposition does not hold because the operators Sat_L, $\Box_i^+$ and $\Diamond_i^+$ may return relations which are not in L-normal labeled form. Also note that L-SPCU queries do not contain "iteration", so in general they are not comparable with recursive L-MDatalog programs.

An additional operator that deserves consideration is the *redundant elimination* operator $RE_L(I) = \{(\triangle, \mathsf{t}) \in I \mid$ there is no $(\triangle', \mathsf{t}) \in I$ such that $\triangle' \neq \triangle$ and $\triangle$ is an L-instance of $\triangle'\}$. We believe that this operator has a good behavior when used in L-SPCU queries.

7 An Approximation Method for L-MDatalog in $L \in \mathcal{UMD}$

Recall that for $L \in \mathcal{UMD}$, the data complexity of L-MDatalog is PSPACE-hard, and the operator Sat_L may return an infinite L-relation even when the input consists of finite L-relations. To overcome these problems we provide an approximation method for evaluating L-MDatalog queries for the case $L \in \mathcal{UMD}$. The idea is to impose a limit on the lengths of modalities that can occur in the computation. In this section, if not stated otherwise, $L \in \mathcal{UMD}$, P is an L-MDatalog program over a schema S, and I is an *edb* instance over *edb*(S) in L.

We use a fixed number $l \geq 1$ as the limit we impose on the lengths of modalities that can occur in the computation of the fixpoint semantics of L-MDatalog programs and L-SPCU queries.

Definition 7.1 Given an L-normal model generator I with modal depth not greater than l, $l\text{-}Sat_L(I)$ is the least extension of I that contains all ground atoms in almost L-normal labeled form that are derivable from some atom in I using the rules specifying Sat_L in the way that no atom in the derivation has modal depth greater than l.

Definition 7.2 Define the operator $l\text{-}T_{L,P}$ analogously as for $T_{L,P}$ but with $l\text{-}Sat_L$ in the place of Sat_L. Define $l\text{-}T_{L,P,I}(J) = l\text{-}T_{L,P}(I \cup J)$. Thus $l\text{-}T_{L,P,I}$ is monotonic and continuous and has the least fixpoint $l\text{-}T_{L,P,I} \uparrow \omega = \bigcup_{0 \leq k \leq \omega} l\text{-}T_{L,P,I} \uparrow k$, where $l\text{-}T_{L,P,I} \uparrow k$ is defined in a similar way as $T_{L,P} \uparrow k$. Let $l\text{-}P_L(I)$ denote the least fixpoint of $l\text{-}T_{L,P,I}$.

Proposition 7.1 *Let P be an L-MDatalog program over a schema S, where $L \in \mathcal{UMD}$, and I be an edb instance over $edb(S)$ in L. Then $l\text{-}P_L(I)$ is an approximation of $P_L(I)$: i) the larger l is the better $l\text{-}P_L(I)$ approximates $P_L(I)$; ii) for every $\alpha \in P_L(I)$, there exists l such that $\alpha \in l\text{-}P_L(I)$. The database instance $l\text{-}P_L(I)$ can be computed in polynomial time and has a polynomial size in the size of I (assuming that P and l are fixed).*

The first assertion of this lemma follows from that $l\text{-}P_L$ is monotonic w.r.t. l. The second assertion can be proved analogously as for Theorem 5.1.

By Proposition 4.2, every L-MDatalog query $(P, \varphi(x_1, \dots, x_k))$ over a schema S can be effectively transformed to an L-MDatalog query $(P', q(x_1, \dots, x_k))$ over $S \cup \{q\}$, where q is an *idb* predicate, such that for every *edb* instance I over *edb*(S) in L and every constant symbols $c_1, \dots, c_k$, $P \cup P_I \models_L \varphi(c_1, \dots, c_k)$ iff $P' \cup P_I \models_L q(c_1, \dots, c_k)$. Thus checking whether $P \cup P_I \models_L \varphi(c_1, \dots, c_k)$ can be done by checking whether $q(c_1, \dots, c_k)$ is true in the standard L-model of $P'_L(I) \cup I$ (by Lemma 4.1), which in turn is equivalent to $q(c_1, \dots, c_k) \in P'_L(I)$. As $l\text{-}P'_L(I)$ is an approximation of $P'_L(I)$, *we can approximate the task of checking whether $P \cup P_I \models_L \varphi(c_1, \dots, c_k)$ by checking whether $q(c_1, \dots, c_k) \in l\text{-}P'_L(I)$, which is solvable in polynomial time in the size of the input I* (assuming that P and l are fixed).

Definition 7.3 Suppose that we are given an L-MDatalog query $(P, q(x_1, \dots, x_k))$ over a schema S, with $q \in idb(S)$, and an *edb* instance I over *edb*(S) in L. Then we call the approximation of the check $P \cup P_I \models_L q(c_1, \dots, c_k)$ by the check $q(c_1, \dots, c_k) \in l\text{-}P_L(I)$ the *l-modal-depth-restricted fixpoint semantics* of the query. Under this semantics, the query $(P, q(x_1, \dots, x_k))$ on the input I returns the set of tuples of constant symbols $(c_1, \dots, c_k)$ such that $q(c_1, \dots, c_k) \in l\text{-}P_L(I)$.

Definition 7.4 We define the algebraic operator $l\text{-}Sat_L$ analogously as for the algebraic operator Sat_L, using the operator acting on model generators $l\text{-}Sat_L$ instead of Sat_L. Define *$L_{|l}$-SPCU algebra* to be the L-SPCU algebra with Sat_L replaced by $l\text{-}Sat_L$. *$L_{|l}$-SPCU queries* are built from input L-relations with modal depth not greater than l and unary constant relations $l\text{-}I^c_L = \{(\boxdot, (c)) \mid \boxdot$ is a universal modality in L-normal labeled form with length not greater than $l\}$, where c is a constant symbol, using the $L_{|l}$-SPCU algebra operators.

The following proposition corresponds to Proposition 6.1 and can be proved analogously.

Proposition 7.2 *Every L-MDatalog query $(P, q(x_1, \dots, x_k))$, where $L \in \mathcal{UMD}$ and P is a nonrecursive L-MDatalog program, is equivalent under the l-modal-depth-restricted fixpoint semantics to an $L_{|l}$-SPCU query.*

8 Seminaive Evaluation of MDatalog

We extend the seminaive evaluation technique of Datalog (see, e.g., [1]) for MDatalog.

First, consider the case $L \in \mathcal{BMD}$. Let P be an L-MDatalog program over a schema S and I an *edb* instance over *edb*(S) in L. We first give a *naive* algorithm for computing $P_L(I)$. Since $P_L(I) = T_{L,P,I} \uparrow \omega$, we can obtain $P_L(I)$ by computing $T_{L,P,I} \uparrow k$ for increasing values of k until a fixpoint $T_{L,P,I} \uparrow k = T_{L,P,I} \uparrow (k-1)$ is reached. Suppose that we have already computed $T_{L,P,I} \uparrow k$ for some k and the content of the relation of an *idb* predicate p in $T_{L,P,I} \uparrow k$ is stored in p_k. Let J_k consist of such relations p_k. To compute $T_{L,P,I} \uparrow (k+1)$ consider the program $P^{(k+1)}$

obtained from P by replacing every *idb* predicate p in bodies of the clauses of P by p_k. $P^{(k+1)}$ is a nonrecursive MDatalog program, and hence $P_L^{(k+1)}(I \cup J_k)$ can be computed using the L-SPCU algebra operators. The results of $P_L^{(k+1)}(I \cup J_k)$ are then assigned to relations p_{k+1} to start the next round (if necessary).

In the naive algorithm, a considerable amount of redundant computation is done, as $T_{L,P,I}\uparrow k \subseteq T_{L,P,I}\uparrow(k+1)$ and each round recomputes all elements of the previous round. To avoid this situation, the seminaive evaluation technique is used. Let $P^{(k+1)'}$, for $k \geq 1$, be the program constructed as follows: for each clause $\boxdot(A \leftarrow B_1, \ldots, B_n)$ of P and each $1 \leq i \leq n$, add to $P^{(k+1)'}$ the clause $\boxdot(A \leftarrow B'_1, \ldots, B'_{i-1}, B^*_i, B''_{i+1}, \ldots, B''_n)$, where B'_j (resp. B''_j) is obtained from B_j by replacing the predicate of B_j, denoted by p, by p_k (resp. p_{k-1}), and B^*_i is obtained from B_i by replacing the predicate of B_i, denoted by q, by the predicate defined by $(q_k - q_{k-1})$. The key in this evaluation is B^*_i, which contains only new atoms that are derived at round k. Then the seminaive algorithm is the modification of the naive algorithm with $P^{(k)}$ replaced by $P^{(k')}$ for $k \geq 2$. It is straightforward to prove that the seminaive algorithm produces $T_{L,P,I}\uparrow k$ at round k. This means that the seminaive algorithm is correct for L-MDatalog with $L \in \mathcal{BMD}$.

Now consider the case $L \in \mathcal{UMD}$. We approximate evaluation of L-MDatalog queries by using the l-modal-depth-restricted fixpoint semantics, where l is a fixed number. Change the text of the case $L \in \mathcal{BMD}$ by replacing P_L by l-P_L and $T_{L,P,I}$ by l-$T_{L,P,I}$. Then the text is still correct. That is, the seminaive algorithm is correct under the l-modal-depth-restricted fixpoint semantics for L-MDatalog with $L \in \mathcal{UMD}$.

9 Top-Down Evaluation of MDatalog

The top-down evaluation method of Datalog closely relates to the standard SLD-resolution calculus of classical logic programming. In this section, basing on our SLD-resolution calculus for MProlog, we extend that method for MDatalog. Apart from techniques involved with modalities, our formulation differs from the framework of annotated query-subquery evaluation presented in the book [1] by Abiteboul et al. in the aspects that we do not use adornments and annotations. With adornments and annotations, the annotated query-subquery evaluation of Datalog executes relational operations only on classical tuples of constant symbols (augmented eventually with annotations). Without adornments and annotations, we do relational operations also on substitutions. The problem is that, adornments and annotations are not sufficient enough to deal with modalities, which may contain variables and atom variables. Besides, they would make the presentation more complicated.

Top-down evaluation of queries simulates SLD-resolution but does it set-at-a-time and manages to find all answers effectively. In SLD-derivations, constant symbols and repeats of variables may be pushed from goals to subgoals through unification, and the search space is thus restricted. Furthermore, "modal contexts" of goal atoms may also be pushed from goals to subgoals. These properties make the top-down evaluation attractive.

In this section, if not stated otherwise, $L \in \mathcal{BMD}$. The case $L \in \mathcal{UMD}$ is considered at the end of this section.

9.1 Informal Description

We first adapt our SLD-resolution calculus for L-MDatalog in order to find all answers effectively. We set up the problem as follows: given an L-MDatalog program over a schema S, an *edb* instance I over *edb*(S) in L, and an atom α in almost L-normal labeled form of an *idb* predicate p, which may contain variables and atom variables, construct an answer L-relation *ans_p* such that for every SLD-refutation of $P \cup P_I \cup \{\leftarrow \alpha\}$ in L with computed answer θ, $\alpha\theta$ is an L-instance of some atom from *ans_p*, treating tuples of *ans_p* as atoms of the predicate p. (Here, θ may contain bindings for atom variables.) This property of *ans_p* is called completeness of the evaluation. We expect also two other properties: soundness and tightness. Soundness states that for every atom α' of *ans_p*, there exists an SLD-refutation of $P \cup P_I \cup \{\leftarrow \alpha'\}$ in L, and tightness informally states that all tuples of *ans_p* closely relate to the main query given for evaluating.

Definition 9.1 A *goal L-tuple* is defined similarly as an L-tuple, except that it may contain variables and atom variables. A *goal L-relation*, also called a *generalized L-relation*, is a set of goal L-tuples of the same arity.

For each *idb* predicate q, we use a global variable ans_q to keep an answer L-relation for q. Tuples of ans_q are treated as atoms of the predicate q. At the beginning, we set all of $ans\_$ variables to empty relations. Consider an SLD-refutation of $P \cup P_I \cup \{\leftarrow \alpha\}$. It begins with a sequence of applications of $rSat_L/rNF_L$ rules and then an application of a program clause of P. Let $\leftarrow \alpha'$ be the result of the application of that sequence of $rSat_L/rNF_L$ rules to $\leftarrow \alpha$ and let the sequence of used mgu's be $\theta_1, \ldots, \theta_j$. Let $\gamma_0 = \theta_1 \ldots \theta_j$. Suppose that the program clause applied to $\leftarrow \alpha'$ is $\varphi = \boxdot(A \leftarrow B_1, \ldots, B_n)$ and the application is as follows:

- $\alpha' = \triangle' A'$, where $\triangle'$ is in L-normal labeled form,
- $\triangle'$ is an L-instance of a universal modality $\boxdot'$ and $\boxdot'(A \leftarrow B_1, \ldots, B_n)$ is an L-instance of the program clause φ,
- θ is an mgu of A' and the forward labeled form of A,
- the new goal is $\leftarrow (\triangle' B_1, \ldots, \triangle' B_n)\theta$.

Let $\delta_0 = \theta$. For each $1 \leq i \leq n$, we process $\leftarrow (\triangle' B_i)\delta_{i-1}$ as follows, where δ_{i-1} is the substitution containing the bindings of variables and atom variables after processing $\leftarrow (\triangle' B_{i-1})\delta_{i-2}$. Let p_i be the predicate of B_i.

1. Case p_i is an *edb* predicate: If $(\triangle' B_i)\delta_{i-1}\gamma_i$ is an L-instance of some atom from $Sat_L(I(p_i))$ for some substitution γ_i then let $\delta_i = \delta_{i-1}\gamma_i$ and continue to process the next goal atom.
2. Case p_i is an *idb* predicate:
   (a) Recursively process $\leftarrow (\triangle' B_i)\delta_{i-1}$ in the same way as for $\leftarrow \alpha$. This task does not pass bindings of variables and atom variables directly outside, but it updates the answer L-relations held by global variables.
   (b) If $(\triangle' B_i)\delta_{i-1}\gamma_i$ is an L-instance of some atom from ans_p_i for some substitution γ_i then let $\delta_i = \delta_{i-1}\gamma_i$ and continue to process the next goal atom.

Then $\gamma_0\delta_n$ holds an answer for $\leftarrow \alpha$. Hence, we would like to add the atom $\alpha\gamma_0\delta_n$ to the answer L-relation ans_p, where p is the predicate of α. Assume that all variables of α occur (also) in the classical atom of α. As P is an range-restricted program, $\alpha\gamma_0\delta_n$ does not contain variables, but it may contain atom variables (in labeled existential modal operators) and this is a problem to solve. Suppose that $\langle X \rangle_j$ occurs in $\alpha\gamma_0\delta_n$. Then $\langle X \rangle_j$ also occurs in α and X is not substituted by $\gamma_0\delta_n$. Observe that we can change $\langle X \rangle_j$ in $\leftarrow \alpha$ at the beginning to $\Box_j$ without affecting the process. Hence, we can change every labeled existential modal operator $\langle X \rangle_j$ in $\alpha\gamma_0\delta_n$ to $\Box_j$ and add the resulting atom to the answer L-relation ans_p. (Another solution is just to standardize the atom variables of $\alpha\gamma_0\delta_n$ as described below and add the obtained atom to ans_p, which is then treated as a generalized L-relation.)

To obtain all answers for the goal $\leftarrow \alpha$, all choices are systematically tried, and the process is repeated until no changes were made to the global $ans\_$ variables during the last iteration of the main loop. To guarantee the stop property, in each iteration of the main loop, each goal like $\leftarrow \alpha$ is processed only once. To do this we standardize α before processing the goal and record the standardized atom in a relation held by a global variable. The standardization is done by renaming variables and atom variables using a fixed list of names, which is disjoint with the list of variables of P and the list of variables and atom variables used for standardization in SLD-derivations, so that if α_1 and α_2 are variants then they have the same standardized atom. The relation containing goal atoms of a predicate p that have been processed is called an input relation and the global variable holding it is named $input\_p$. It can be represented as a goal L-relation and we treat tuples of $input\_p$ as atoms of the predicate p. The global $input\_$ variables are reset to empty L-relations for each iteration of the main loop.

Here are some remarks to our adaptation of SLD-resolution for MDatalog:

- We concentrate on goals with a single atom rather than goals with more atoms.
- Standardizing variables is done for goal atoms but not for input program clauses and is aimed also to avoid redundant repeated computations.
- When processing the body of a program clause, the modality $\triangle'$ together with the bindings of variables and atom variables is passed from left to right (rather than top-down from goal to subgoal).
- In the step 1, we find an answer γ_i for $P_I \cup \{\leftarrow (\triangle' B_i)\delta_{i-1}\}$ by checking whether $(\triangle' B_i)\delta_{i-1}\gamma_i$ is an L-instance of some atom from $Sat_L(I)$. (Note that I may be given without P_I.) This is supported by the following lemma.

Lemma 9.1 *Let P be an L-MDatalog program over a schema S with $edb(S) = \emptyset$, where $L \in \mathcal{BMD} \cup \mathcal{UMD}$, and $\leftarrow \triangle A$ be a goal where $\triangle$ is in L-normal labeled form and A does not contain any labeled existential modal operator. Let θ be a computed answer in L of $P \cup \{\leftarrow \triangle A\}$. Then $(\triangle A)\theta$ is an L-instance of some atom from $Sat_L(I_{L,P})$. (Recall that $I_{L,P} = T_{L,P}\uparrow\omega$.)*

Sketch. Let M be the standard L-model graph of $I_{L,P}$, σ_0 be the standard $\Diamond$-realization function on M, and σ be a maximal $\Diamond$-realization function on M that extends σ_0. By the proof of Lemma 5.15 of [25], there exists a $\Box$-lifting form $\triangle' A$ of $\triangle A$ such that $M, \sigma \vDash \forall_c((\triangle' A)\theta)$. Since P is an range-restricted program, the constant symbols used in $I_{L,P}$ and for the construction of M are the constant symbols occurring in P. Assuming that the signature contains some other constant symbols, it follows from $M, \sigma \vDash \forall_c((\triangle' A)\theta)$ that $(\triangle' A)\theta$ does not contain variables (but may contain atom variables). Let $\triangle''$ be the instance of $\triangle'\theta$ obtained by replacing each atom variable by $\top$ and each modal operator $\Box_i$ by $\langle\top\rangle_i$. We have that $M, \sigma \vDash \triangle'' A\theta$. Hence, for the possible world $w = \triangle''$, we have that $M, w \vDash A\theta$.

Let M' be the *extended L-model graph of $I_{L,P}$*, which is defined similarly as the standard L-model graph of $I_{L,P}$ except that $Sat_L(I_{L,P})$ is used instead of $Ext_L(I_{L,P})$. Recall that $Ext_L(I_{L,P}) \subseteq Sat_L(I_{L,P})$. Observe that M and M' have the same frame (because if $\triangle^\dagger\langle E\rangle_i\alpha \in Sat_L(I_{L,P})$ and $\triangle^\dagger$ does not contain any unlabeled existential modal operator then $\triangle^\dagger\langle E\rangle_i\alpha' \in Ext_L(I_{L,P})$ for some α'). Furthermore, for every possible world w of M and M', the content $H(w)$ of w in M is a subset of the content $H'(w)$ of w in M', and $H'(w) - H(w)$ consists of atoms with some unlabeled existential modal operators.

Since $M, w \vDash A\theta$, it can be shown that $A\theta$ is an L-instance of some atom from $H'(w)$. For example, consider the case $L = KD45_{(m)}$. There are 3 subcases:

- Case $A\theta = E$: Since $M, w \vDash A\theta$, it is clear that $A\theta \in H'(w)$.
- Case $A\theta = \Box_i E$: Consider the case $w\langle\top\rangle_i$ is a modality in L-normal labeled form. Since $M, w \vDash A\theta$, we have that $E \in H'(w\langle\top\rangle_i)$. It follows that $w\langle\top\rangle_i E$ is an L-instance of some atom from $Sat_L(I_{L,P})$. Hence $\Box_i E$ is an L-instance of some atom from $H'(w)$. Now consider the case $w\langle\top\rangle_i$ is not a modality in L-normal labeled form. We have that $w = u\langle F\rangle_i$ for some u and F. Since $M, w \vDash A\theta$, we have that $E \in H'(u\langle\top\rangle_i)$. It follows that $u\langle\top\rangle_i E$ is an L-instance of some atom from $Sat_L(I_{L,P})$. Hence, $u\Box_i E$ and $u\Box_i\Box_i E$ are L-instances of some atoms from $Sat_L(I_{L,P})$. Hence $\Box_i E$ is an L-instance of some atom from $H'(w)$.
- Case $A\theta = \Diamond_i E$: Since $M, w \vDash A\theta$, either i) $\langle F\rangle_i E \in H'(w)$ for some F or ii) $E \in H'(u)$, $R_i(w, u)$ holds, $\sigma(v, \langle F\rangle_i) = w$, and $\sigma(v, \langle F'\rangle_i) = u$ for some u, v, F, F'. Consider the second case. Since $\sigma(v, \langle F'\rangle_i) = u$ and $E \in H'(u)$, either $\langle F'\rangle_i E \in H'(v)$ or $\Box_i E \in H'(v)$. Hence $v\langle F'\rangle_i E$ or $v\Box_i E$ is an L-instance of some atom of $Sat_L(I_{L,P})$. It follows that $v\Box_i\Diamond_i E$ or $v\Box_i\Box_i E$ is an L-instance of some atom of $Sat_L(I_{L,P})$. Hence $\Diamond_i E \in H'(w)$ or $\Box_i E \in H'(w)$.

Since $A\theta$ is an L-instance of some atom from $H'(w)$ and $\triangle'' = w$, $\triangle'' A\theta$ is an L-instance of some atom from $Sat_L(I_{L,P})$, which implies that $(\triangle A)\theta$ is also an L-instance of some atom from $Sat_L(I_{L,P})$. •

9.2 A Formal Presentation of the Algorithm

We now formally present the algorithm of the evaluation method described in the previous section.

Algorithm 9.1

Evaluate an L-MDatalog query $(P, q(x_1, \ldots, x_k))$ over a schema S, where $L \in \mathcal{BMD}$ and $q \in idb(S)$, on an *edb* instance I over $edb(S)$ in L.

1. Initialize the global variables ans_p to empty L-relations for every $p \in idb(S)$.
2. Repeat
   (a) set the global variables $input\_p$ to empty L-relations for every $p \in idb(S)$
   (b) call Procedure 9.2 to process the goal $\leftarrow q(x_1, \ldots, x_k)$

   until the global *ans_* variables were not changed during the last iteration.
3. Return $\{\mathsf{t} \mid (\centerdot, \mathsf{t}) \in ans\_q$ where $\centerdot$ is the empty modality$\}$.

Procedure 9.2

Process a goal $\leftarrow \alpha$, where α is an atom in almost L-normal labeled form.

1. Standardize variables and atom variables of α. Let the resulting atom be α'.
2. Let p be the predicate of α'. If the goal L-tuple representing α' already belongs to $input\_p$ then exit, else add the tuple to $input\_p$.
3. For each program clause φ defining p in P: Call Procedure 9.3 to process the goal $\leftarrow \alpha'$ on φ.

Procedure 9.3

Process a goal $\leftarrow \alpha$ on a program clause $\varphi = \boxdot(A \leftarrow B_1, \ldots, B_n)$, where α is a standardized atom in almost L-normal labeled form and has the same predicate as A.

1. If the classical atoms of α and A cannot be unified then exit.
2. For every atom α' derivable from α using a sequence of applications of $rSat_L/rNF_L$ rules and a sequence of mgu's $\theta_1, \ldots, \theta_j$, do:
   - Let $\gamma_0 = \theta_1 \ldots \theta_j$ and let $\alpha' = \triangle' A'$, where A' and A have modalities of the same length.
   - Let $\boxdot'$ be the universal modality that is a $\Box$-lifting of $\triangle'$. If $\boxdot'(A \leftarrow B_1, \ldots, B_n)$ is an L-instance of the program clause φ, and δ_0 is an mgu of A' and the forward labeled form of A, then:
     (a) $sup_0 := \{\delta_0\}$.[13]
     (b) For each i from 1 to n do:
       i. Let p_i be the predicate of B_i.
       ii. $sup_i := \emptyset$.
       iii. Case $p_i \in edb(S)$: For every $\delta_{i-1} \in sup_{i-1}$ and every atom $\beta \in Sat_L(I(p_i))$, if $(\triangle' B_i)\delta_{i-1}\gamma_i$ is an L-instance of β using a most general substitution γ_i then add $\delta_{i-1}\gamma_i$ to sup_i.
       iv. Case $p_i \in idb(S)$: For every $\delta_{i-1} \in sup_{i-1}$ do:
         A. Call Procedure 9.2 to process the goal $\leftarrow (\triangle' B_i)\delta_{i-1}$.
         B. For every atom $\beta \in ans\_p_i$, if $(\triangle' B_i)\delta_{i-1}\gamma_i$ is an L-instance of β using a most general substitution γ_i then add $\delta_{i-1}\gamma_i$ to sup_i.
     (c) For each $\delta_n \in sup_n$ do: let α'' be the atom obtained from $\alpha\gamma_0\delta_n$ by replacing every modal operator of the from $\langle X \rangle_j$ by $\Box_j$; add the L-tuple representing α'' to the L-relation ans_p, where p is the predicate of α.

[13] sup_i denotes ith "supplementary" relation.

9.3 Properties of the Algorithm

In this subsection, we prove that the top-down evaluation method for MDatalog presented by Algorithm 9.1 is sound, complete, and tight. Roughly speaking, Algorithm 9.1 is a reformulation of SLD-resolution for MDatalog with a different way of passing bindings of variables and atom variables. Our proofs are therefore based on the proofs of soundness and completeness of our SLD-resolution calculus for MProlog.

Theorem 9.2 (Soundness) *Let $(P, q(x_1, \dots, x_k))$ be an L-MDatalog query over a schema S, where $L \in \mathcal{BMD}$, and I be an edb instance over $edb(S)$ in L. Consider the execution of Algorithm 9.1 for that query on I. Then, for every $\alpha'' \in ans\_p$, $P \cup P_I \cup \{\leftarrow \alpha''\}$ has an SLD-refutation in L.*

Proof. We prove this theorem by induction on the time when α'' is added to *ans_p* in Step 2c of Procedure 9.3. Let $\triangle''$ be the modality obtained from $\triangle'\delta_n$ by replacing every modal operator of the from $\langle X \rangle_j$ by $\Box_j$ (where $\triangle'$ and δ_n are the objects used in Procedure 9.3).

We can simulate the refutation of $\leftarrow \alpha$ embedded in Procedure 9.3 for $\leftarrow \alpha''$ as follows. Instead of the goal $\leftarrow (\triangle' B_1, \dots, \triangle' B_n)\delta_0$ (or the sequence of goals $\leftarrow (\triangle' B_1)\delta_0, \dots, \leftarrow (\triangle' B_n)\delta_{n-1}$), we have the ground goal $\leftarrow \triangle'' B_1 \delta_n, \dots, \triangle'' B_n \delta_n$.

Consider Step 2(b)iii of Procedure 9.3. We have that $\triangle'' B_i \delta_n$ is an L-instance of $\beta \in Sat_L(I(p_i))$. By Corollary 5.20 of [25], $P_I \cup \{\leftarrow \beta\}$ has an SLD-refutation in L. Hence $P \cup P_I \cup \{\leftarrow \triangle'' B_i \delta_n\}$ also has an SLD-refutation in L.

Consider Step 2(b)ivB of Procedure 9.3. We have that $\triangle'' B_i \delta_n$ is an L-instance of $\beta \in ans\_p_i$. By the inductive assumption, $P \cup P_I \cup \{\leftarrow \beta\}$ has an SLD-refutation in L. Hence $P \cup P_I \cup \{\leftarrow \triangle'' B_i \delta_n\}$ also has an SLD-refutation in L.

The refutations of $P \cup P_I \cup \{\leftarrow \triangle'' B_i \delta_n\}$ for $1 \le i \le n$ can be combined into an SLD-refutation of $P \cup P_I \cup \{\leftarrow \triangle'' B_1 \delta_n, \dots, \triangle'' B_n \delta_n\}$ in L because the goal is ground. (That is, for i from 1 to $n-1$, we apply the refutation for $\leftarrow \triangle'' B_i \delta_n$ to reduce the goal $\leftarrow \triangle'' B_i \delta_n, \dots, \triangle'' B_n \delta_n$ to $\leftarrow \triangle'' B_{i+1} \delta_n, \dots, \triangle'' B_n \delta_n$.) •

We say that an SLD-derivation uses the *left-most-atom selection function* if the selected atom of each goal in the derivation is the left most atom of the goal.

We need the following lemma for Completeness Theorem 9.4.

Lemma 9.3 *Let $(P, q(x_1, \dots, x_k))$ be an L-MDatalog query over a schema S, where $L \in \mathcal{BMD}$, and I be an edb instance over $edb(S)$ in L. Consider the end moment of an execution of Algorithm 9.1 for that query on I. Let $p \in idb(S)$, $(\triangle, \mathfrak{t}) \in input\_p$, and let θ be the computed answer of an SLD-refutation of $P \cup P_I \cup \{\leftarrow \triangle p(\mathfrak{t})\}$ in L that uses the left-most-atom selection function. Then ans_p contains some $\Box$-lifting form of $(\triangle p(\mathfrak{t}))\theta$ (here, ans_p is treated as a set of atoms of the predicate p).*

Proof. We prove this lemma by induction on the length of the mentioned SLD-refutation. Let $\theta_1, \dots, \theta_h$ be the sequence of mgu's used in the refutation. We have that $(\triangle p(\mathfrak{t}))\theta_1 \dots \theta_h = (\triangle p(\mathfrak{t}))\theta$. Let $\alpha = \triangle p(\mathfrak{t})$ and suppose that the first fragment of the refutation of $\leftarrow \alpha$ uses a sequence of applications of $rSat_L/rNF_L$ rules with mgu's $\theta_1, \dots, \theta_j$, resulting in a goal $\alpha' = \Delta' A'$, and then uses a variant $\varphi' = \boxdot(A'' \leftarrow B'_1, \dots, B'_n)$ of a program clause $\varphi = \boxdot(A \leftarrow B_1, \dots, B_n)$ of P with mgu θ_{j+1}, resulting in the goal $\leftarrow (\triangle' B'_1, \dots, \triangle' B'_n)\theta_{j+1}$. We have that $\varphi' = \varphi\varrho$ (i.e., $A'' = A\varrho$ and $B'_i = B_i\varrho$ for $1 \le i \le n$) for some renaming substitution ϱ that uses only variables of φ and φ', and that θ_{j+1} is an mgu of A' and the forward labeled form of A''. Let $j_1 = j+2$, $j_{n+1} = h+1$ and suppose that the fragment for processing $\leftarrow (\triangle' B'_i)\theta_{j_1 - 1}\theta_{j_1} \dots \theta_{j_i - 1}$ of the refutation of $\leftarrow \alpha$ uses mgu's $\theta_{j_i}, \dots, \theta_{j_{i+1}-1}$. Thus, after processing the atom B'_{i-1}, for $2 \le i \le n+1$, the next goal of the refutation of $\leftarrow \alpha$ is $\leftarrow (\triangle' B'_i, \dots, \triangle' B'_n)\theta_{j_1 - 1}\theta_{j_1} \dots \theta_{j_i - 1}$.

Consider the last iteration of the main loop of Algorithm 9.1, the execution of Procedure 9.2 at which $(\triangle, \mathfrak{t})$ is added into *input_p* (in that iteration), and the processing of the goal $\leftarrow \alpha$ on the program clause φ by Procedure 9.3 (in that execution of Procedure 9.2).

We have that θ_{j+1} is an mgu of A' and the forward labeled form of $A\varrho$ (since $A'' = A\varrho$). As ϱ does not use variables and atom variables of A', we have that $A' = A'\varrho$. Hence θ_{j+1} is an mgu of $A'\varrho$ and the forward labeled form of $A\varrho$. Since δ_0 is an mgu of A' and the forward labeled form of A, it follows that $\varrho\theta_{j+1} = \delta_0\gamma'_0$ for some substitution γ'_0.

As an inner induction, let the inductive hypothesis be that after processing the atom B_{i-1} by Procedure 9.3, where $2 \leq i \leq n+1$, or at the beginning if $i = 1$, it holds that $\varrho\theta_{j_1-1}\theta_{j_1}\ldots\theta_{j_i-1} = \delta_{i-1}\gamma'_{i-1}$ for some $\delta_{i-1} \in sup_{i-1}$ and some γ'_{i-1}. This inductive hypothesis clearly holds for $i = 1$ (since $j_1 = j+2$ and $\varrho\theta_{j+1} = \delta_0\gamma'_0$). Suppose that the inductive hypothesis holds for some $1 \leq i \leq n$. We show that it also holds for $i+1$.

Since $\varrho\theta_{j_1-1}\theta_{j_1}\ldots\theta_{j_i-1} = \delta_{i-1}\gamma'_{i-1}$ and ϱ does not use variables and atom variables of α, α' and Δ', we have that:

$$\begin{array}{rl} & \leftarrow (\triangle' B'_i)\theta_{j_1-1}\theta_{j_1}\ldots\theta_{j_i-1} \\ = & \leftarrow (\triangle'(B_i\varrho))\theta_{j_1-1}\theta_{j_1}\ldots\theta_{j_i-1} \\ = & \leftarrow (\triangle' B_i)\varrho\theta_{j_1-1}\theta_{j_1}\ldots\theta_{j_i-1} \\ = & \leftarrow (\triangle' B_i)\delta_{i-1}\gamma'_{i-1} \end{array}$$

Since there is a refutation of $\leftarrow (\triangle' B'_i)\theta_{j_1-1}\theta_{j_1}\ldots\theta_{j_i-1}$ using mgu's $\theta_{j_i},\ldots,\theta_{j_{i+1}-1}$, by Lifting Lemma 5.18 of [25], there exists a refutation of $\leftarrow (\triangle' B_i)\delta_{i-1}$ using mgu's $\theta'_{j_i},\ldots,\theta'_{j_{i+1}-1}$ such that $\gamma'_{i-1}\theta_{j_i}\ldots\theta_{j_{i+1}-1} = \theta'_{j_i},\ldots,\theta'_{j_{i+1}-1}\mu_i$ for some μ_i.

Consider the case when the predicate p_i of B_i is an *edb* predicate. By Lemma 9.1, $(\triangle' B_i)\delta_{i-1}\theta'_{j_i}\ldots\theta'_{j_{i+1}-1}$ is an L-instance of some atom $\beta \in Sat_L(I(p_i))$. Hence, there exists a most general substitution γ_i such that $(\triangle' B_i)\delta_{i-1}\gamma_i$ is an L-instance of β and $\gamma_i\mu'_i = \theta'_{j_i},\ldots,\theta'_{j_{i+1}-1}$ for some substitution μ'_i. Let $\gamma'_i = \mu'_i\mu_i$. We have that:

$$\begin{array}{rl} & \varrho\theta_{j_1-1}\ldots\theta_{j_{i+1}-1} \\ = & (\varrho\theta_{j_1-1}\ldots\theta_{j_i-1})\theta_{j_i}\ldots\theta_{j_{i+1}-1} \\ = & \delta_{i-1}\gamma'_{i-1}\theta_{j_i}\ldots\theta_{j_{i+1}-1} \\ = & \delta_{i-1}\theta'_{j_i}\ldots\theta'_{j_{i+1}-1}\mu_i \\ = & \delta_{i-1}\gamma_i\mu'_i\mu_i \\ = & \delta_{i-1}\gamma_i\gamma'_i. \end{array}$$

Hence, for $\delta_i = \delta_{i-1}\gamma_i \in sup_i$, we have that $\theta_{j_1-1}\ldots\theta_{j_{i+1}-1} = \delta_i\gamma'_i$. That is, the inductive hypothesis of the second induction holds for $i+1$.

Consider the case when the predicate p_i of B_i is an *idb* predicate. Let $(\triangle' B_i)\delta_{i-1}\rho$ be the standardized atom of $(\triangle' B_i)\delta_{i-1}$, where ρ is a renaming substitution. Since there exists a refutation of $\leftarrow (\triangle' B_i)\delta_{i-1}(\rho\rho^{-1})$ (i.e., $\leftarrow (\triangle' B_i)\delta_{i-1}$) using mgu's $\theta'_{j_i},\ldots,\theta'_{j_{i+1}-1}$, by Lifting Lemma 5.18 of [25], there exists a refutation of $\leftarrow (\triangle' B_i)\delta_{i-1}\rho$ using mgu's $\theta''_{j_i},\ldots,\theta''_{j_{i+1}-1}$ such that $\rho^{-1}\theta'_{j_i},\ldots,\theta'_{j_{i+1}-1} = \theta''_{j_i},\ldots,\theta''_{j_{i+1}-1}\rho'$ for some substitution ρ'. As Procedure 9.2 is called for $\leftarrow (\triangle' B_i)\delta_{i-1}$, the goal L-tuple representing $(\triangle' B_i)\delta_{i-1}\rho$ belongs to *input_p*$_i$. By the inductive assumption of the outer induction, some $\Box$-lifting form β of $(\triangle' B_i)\delta_{i-1}\rho\theta''_{j_i},\ldots,\theta''_{j_{i+1}-1}$ belongs to *ans_p*$_i$. Since $(\triangle' B_i)\delta_{i-1}\rho\theta''_{j_i},\ldots,\theta''_{j_{i+1}-1}\rho'$ is an L-instance of β, and $\rho^{-1}\theta'_{j_i},\ldots,\theta'_{j_{i+1}-1} = \theta''_{j_i},\ldots,\theta''_{j_{i+1}-1}\rho'$, we have that $(\triangle' B_i)\delta_{i-1}\rho\rho^{-1}\theta'_{j_i},\ldots,\theta'_{j_{i+1}-1}$ is also an L-instance of β. Hence, there exists a most general substitution γ_i such that $(\triangle' B_i)\delta_{i-1}\gamma_i$ is an L-instance of β and $\gamma_i\mu'_i = \theta'_{j_i},\ldots,\theta'_{j_{i+1}-1}$ for some substitution μ'_i. Analogously as for the case when p_i is an *edb* predicate, for $\gamma'_i = \mu'_i\mu_i$ and $\delta_i = \delta_{i-1}\gamma_i \in sup_i$, the inductive hypothesis of the inner induction holds for $i+1$. This completes the proof of the inner induction.

By the inner induction, we have that $\varrho\theta_{j_1-1}\ldots\theta_{j_{n+1}-1} = \delta_n\gamma'_n$. That is, $\varrho\theta_{j+1}\ldots\theta_h = \delta_n\gamma'_n$. Hence $\theta_1\ldots\theta_h = \gamma_0\varrho^{-1}\delta_n\gamma'_n$. Since ϱ^{-1} does not use variables and atom variables of $\alpha\gamma_0$, we have that $\alpha\gamma_0\varrho^{-1}\delta_n\gamma'_n = \alpha\gamma_0\delta_n\gamma'_n$. The atom α'' added to *ans_p* in Step 2c of Procedure 9.3 is a $\Box$-lifting form of $\alpha\gamma_0\delta_n$ and is ground, so it is also a $\Box$-lifting form of $\alpha\gamma_0\delta_n\gamma'_n = \alpha\gamma_0\varrho^{-1}\delta_n\gamma'_n = (\triangle p(\mathrm{t}))\theta_1\ldots\theta_h = (\triangle p(\mathrm{t}))\theta$. ●

Theorem 9.4 (Completeness) *The execution of Algorithm 9.1 for an L-MDatalog query $(P, q(x_1,\ldots,x_k))$ on an edb instance I returns a tuple $(x_1,\ldots,x_k)\theta$ for every correct answer θ in L of $P \cup P_I \cup \{\leftarrow q(x_1,\ldots,x_k)\}$.*

Proof. Since SLD-resolution for L-MProlog is complete and P, P_I are range-restricted, θ must be a computed answer in L of $P \cup P_I \cup \{\leftarrow q(x_1,\ldots,x_k)\}$. Furthermore, as SLD-resolution for L-MProlog is "strongly complete" [27] (i.e. complete when any "selection function" is used), we can assume that the refutation that gives the computed answer uses the left-most-atom selection function. By Lemma 9.3, $q(x_1, \ldots, x_k)\theta$ is added to *ans_q*, which implies the assertion of the theorem. •

Definition 9.2 An *unrestricted SLD-derivation* in L is an SLD-derivation in L, except that we drop the requirement that the used substitutions θ_i are most general unifiers. They are only required to be unifiers. In an unrestricted SLD-derivation, if a goal G_i is derived from G_{i-1} and an $rSat_L$ rule variant, then θ_i can be arbitrary and $G_i = G'_i\theta_i$, where G'_i is the goal derived from G_{i-1} and that $rSat_L$ rule variant in the usual way.

Theorem 9.5 (Tightness) *Let $(P, q(x_1, \ldots, x_k))$ be an L-MDatalog query over a schema S, where $L \in \mathcal{BMD}$, and I be an edb instance over $edb(S)$ in L. Consider the result of the execution of Algorithm 9.1 for that query on I. Then:*

1. *For every $p \in idb(S)$ and every atom $\alpha' \in input\_p$, there is a variant α of α' that appears in an unrestricted SLD-derivation from $P \cup P_I \cup \{\leftarrow q(x_1, \ldots, x_k)\}$ in L.*

2. *For every $p \in idb(S)$ and every atom $\alpha'' \in ans\_p$, there exists $\alpha \in input\_p$ such that α'' is a $\Box$-lifting form of $\alpha\theta$ for some θ and $P \cup P_I \cup \{\leftarrow \alpha''\}$ has an SLD-refutation in L.*

The first assertion states that every *input_* atom closely relates to the given query, while the second assertion states that every *ans_* atom closely relates to some *input_* atom, and therefore closely relates to the given query.

Proof. Consider the first assertion. For convenience, we rewrite this assertion to: for every $p_i \in idb(S)$ and every atom $\beta'_i \in input\_p_i$, there is a variant β_i of β'_i that appears in an unrestricted SLD-derivation from $P \cup P_I \cup \{\leftarrow q(x_1, \ldots, x_k)\}$ in L. To prove this, it suffices to consider Step 2(b)ivA of Procedure 9.3, which adds a variant of $(\triangle' B_i)\delta_{i-1}$ to $input\_p_i$, and show that $(\triangle' B_i)\delta_{i-1}$ appears in an unrestricted SLD-derivation from $P \cup P_I \cup \{\leftarrow \alpha\}$ in L (where $\leftarrow \alpha$ is the input of Procedure 9.3).

We can start from the goal $\leftarrow (\triangle' B_1, \ldots, \triangle' B_n)\delta_0$, which is derived from $\leftarrow \alpha$. It suffices to show that for every $1 \leq j < i$, the goal $\leftarrow (\triangle' B_{j+1}, \ldots, \triangle' B_n)\delta_j$ is derivable from $\leftarrow (\triangle' B_j, \ldots, \triangle' B_n)\delta_{j-1}$ using an unrestricted SLD-derivation in L, where δ_{i-1} is formed as $\delta_0\gamma_1 \ldots \gamma_{i-1}$ and $\delta_k = \delta_{k-1}\gamma_k$ for every $1 \leq k \leq i-1$.

Consider the case when the predicate p_j of B_j is an *edb* predicate. Let $\beta \in Sat_L(I(p_j))$ be the atom mentioned in Step 2(b)iii of Procedure 9.3. By Corollary 5.20 of [25], $P_I \cup \{\leftarrow \beta\}$ has an SLD-refutation in L. Since $(\triangle' B_j)\delta_{j-1}\gamma_j$ is an L-instance of the ground atom β, there exists an SLD-resolution of $P_I \cup \{\leftarrow (\triangle' B_j)\delta_{j-1}\gamma_j\}$ in L with the empty computed answer. Hence the goal $\leftarrow (\triangle' B_{j+1}, \ldots, \triangle' B_n)\delta_{j-1}\gamma_j$, which is equivalent to $\leftarrow (\triangle' B_{j+1}, \ldots, \triangle' B_n)\delta_j$, is derivable from $\leftarrow (\triangle' B_j, \ldots, \triangle' B_n)\delta_{j-1}$ using an unrestricted SLD-derivation in L.

Consider the case when the predicate p_j of B_j is an *idb* predicate. Let $\beta \in ans\_p_j$ be the atom mentioned in Step 2(b)ivB of Procedure 9.3. By Theorem 9.2, $P \cup P_I \cup \{\leftarrow \beta\}$ has an SLD-refutation in L. Analogously as for the case when p_j is an *edb* predicate, we can derive $\leftarrow (\triangle' B_{j+1}, \ldots, \triangle' B_n)\delta_j$ from $\leftarrow (\triangle' B_j, \ldots, \triangle' B_n)\delta_{j-1}$ using an unrestricted SLD-derivation in L. This completes the proof of the first assertion.

The second assertion follows from Step 2c of Procedure 9.3, with $\theta = \gamma_0\delta_n$. For this, note that when Procedure 9.3 is called for $\leftarrow \alpha$, we have that $\alpha \in input\_p$. Besides, by Theorem 9.2, $P \cup P_I \cup \{\leftarrow \alpha''\}$ has an SLD-refutation in L. •

9.4 Doing It Set-at-a-Time

Operations for databases are often done set-at-a-time instead of tuple-at-a-time. This approach allows various optimizations, for example, sorting, indexing, clustering, etc. In this subsection, we reformulate Algorithm 9.1 using the set-at-a-time technique. For the new algorithm, we use the following relational operators:

- *standardize*(J) for a goal L-relation J returns the goal L-relation consisting of standardized tuples of J.

- $rSatNF_L(J)$ for a goal L-relation J returns the relation consisting of tuples $(\alpha, \gamma_0, \alpha')$ for each $\alpha \in J$ and each SLD-derivation of $\leftarrow \alpha'$ from $\leftarrow \alpha$ using a sequence of $rSat_L/rNF_L$ rules and a sequence of mgu's $\theta_1, \ldots, \theta_j$, where γ_0 is the restriction of $\theta_1 \ldots \theta_j$ to the set of variables and atom variables of α. If (γ_0, α') and (γ_0', α'') are variants (i.e. identical up to a renaming substitution) then we treat $(\alpha, \gamma_0, \alpha')$ and $(\alpha, \gamma_0', \alpha'')$ as the same tuple. For this, some standardization can be done during the derivations. Under this assumption, for $L \in \mathcal{BMD}$, $rSatNF_L(J)$ is a finite relation that can be computed in linear time.
- $resolve(K, \boxdot, A)$
  - where K has the format as $rSatNF_L(J)$ for some J, $\boxdot$ and A are the modal context and the head of an L-MDatalog program clause, respectively,
  - returns the relation consisting of tuples $(\alpha\gamma_0, \triangle', \delta_0)$ for each tuple $(\alpha, \gamma_0, \alpha') \in K$ such that $\alpha' = \triangle' A'$, the universal modality being a $\Box$-lifting form of $\triangle'$ is an L-context instance of $\boxdot$, and δ_0 is the mgu of A' and the forward labeled form of A.
- $resolve\_subgoal(K', B_i, R)$
  - where K' has the format as $resolve(K, \boxdot, A)$ for some K, $\boxdot$, A; B_i is an atom of the form $\Box_j E$, $\Diamond_j E$, or E, with E being a classical atom; and R is an L-relation of the predicate of E,
  - returns the set of tuples $(\alpha\gamma_0, \triangle', \delta_i)$ with $\delta_i = \delta_{i-1}\gamma_i$ for each $(\alpha\gamma_0, \triangle', \delta_{i-1}) \in K'$ and a most general substitution γ_i such that $(\triangle' B_i)\delta_{i-1}\gamma_i$ is an L-instance of some atom from R.

Here is our reformulation of Algorithm 9.1 :

Algorithm 9.4
Evaluate an L-MDatalog query $(P, q(x_1, \ldots, x_k))$ over a schema S, where $L \in \mathcal{BMD}$ and $q \in idb(S)$, on an *edb* instance I over $edb(S)$ in L.

1. Initialize the global variables ans_p to empty L-relations for every $p \in idb(S)$.
2. Repeat
   (a) set the global variables $input\_p$ to empty L-relations for every $p \in idb(S)$.
   (b) call Procedure 9.5 to process the goal L-relation consisting of the atom $q(x_1, \ldots, x_k)$

   until the global $ans\_$ variables were not changed during the last iteration.
3. Return $\{\mathsf{t} \mid (\centerdot, \mathsf{t}) \in ans\_q$ where $\centerdot$ is the empty modality$\}$.

Procedure 9.5
Process a goal L-relation J of a predicate p.

1. $J := standardize(J)$.
2. $J := J - input\_p$.
3. Exit if J is empty.
4. $input\_p := input\_p \cup J$.
5. $K := rSatNF_L(J)$.
6. For each program clause φ defining p in P :
   call Procedure 9.5 to process the goal L-relation K using φ.

Procedure 9.6
Process a goal L-relation K of a predicate p using a program clause $\varphi = \boxdot(A \leftarrow B_1, \ldots, B_n)$ of P.

```
1. $K' := resolve(K, \boxdot, A)$.
2. $i := 0$.
3. While $i < n$ and $K'$ is not empty do:
   (a) $i := i + 1$.
   (b) If the predicate $p_i$ of $B_i$ is an $edb$ predicate then:
       $K' := resolve\_subgoal(K', B_i, Sat_L(I(p_i)))$;
   (c) Else (the predicate $p_i$ of $B_i$ is an $idb$ predicate):
       i. Recursively call Procedure 9.5 for $\{(\triangle' B_i)\delta_{i-1} \mid (\alpha\gamma_0, \triangle', \delta_{i-1}) \in K'\}$.
       ii. $K' := resolve\_subgoal(K', B_i, ans\_p_i)$.
4. $ans\_p := ans\_p \cup \{\alpha'' \mid (\alpha\gamma_0, \triangle', \delta_n) \in K'$ and $\alpha''$ is the atom obtained from $\alpha\gamma_0\delta_n$ by replacing
   every modal operator of the from $\langle X \rangle_j$ by $\Box_j\}$.
```

Theorem 9.6 *The evaluation by Algorithm 9.4 is sound, complete, and tight (i.e., Theorems 9.2, 9.4, 9.5 still hold when "Algorithm 9.1" is replaced by "Algorithm 9.4").*

Sketch. Algorithm 9.4 simulates Algorithm 9.1 but the order of calls of simulations of Procedure 9.2 (by using Procedure 9.5) is different. However, this difference does not affect the adaptation of the proofs of Algorithm 9.1 for Algorithm 9.4. •

Theorem 9.7 *The data complexity of Algorithm 9.4 is in PTIME. (Here, the data complexity is the time complexity measured in the size of the edb instance when the query is fixed.)*

Sketch. Analogously as for the proof of Theorem 5.1 on the data complexity of L-MDatalog, one of the keys here is that modal depths of atoms appearing in *ans_* and *input_* relations are bounded by a constant. Another key is that tuples of *input_* relations are standardized and the operator $rSatNF_L$ can be computed in polynomial time. •

9.5 Further Optimizations

We informally mention further optimizations for Algorithm 9.4:

- Step 4 of Procedure 9.5 is better done in the way so that the answer relation ans_p does not contain any pair of different atoms α_1, α_2 such that α_1 is an L-instance of α_2.
- In Algorithm 9.4, a standardized goal atom is processed only if it does not belong to the corresponding *input_* relation. To increase efficiency, we can process a standardized goal atom only if it is not an L-instance of a fresh variant of any atom from the corresponding *input_* relation. Step 2 of Procedure 9.5 can be modified accordingly.
- The relational operator $resolve\_subgoal$ can be optimized by restricting the substitutions δ_i in the returned tuples $(\alpha\gamma_0, \triangle', \delta_i)$ to the set of "essential" variables and atom variables as follows. Let $resolve\_subgoal$ have an additional parameter V, which for the calls of $resolve\_subgoal$ in Step 3 of Procedure 9.5 is the set of all variables occurring in $(B_{i+1}, \ldots, B_n)$. Then $resolve\_subgoal(K', B_i, R, V)$ returns the set of tuples $(\alpha\gamma_0, \triangle', \delta_{i|V \cup Var(\alpha\gamma_0) \cup Var(\triangle')})$ with $\delta_i = \delta_{i-1}\gamma_i$ for each $(\alpha\gamma_0, \triangle', \delta_{i-1}) \in K'$ and a most general substitution γ_i such that $(\triangle' B_i)\delta_{i-1}\gamma_i$ is an L-instance of some atom from R. Here, $Var(\alpha\gamma_0)$ (resp. $Var(\triangle')$) denotes the set of all variables and atom variables occurring in $\alpha\gamma_0$ (resp. $\triangle'$).

9.6 The Case $L \in \mathcal{UMD}$

Similarly as for the l-modal-depth-restricted fixpoint semantics, we can adjust the top-down evaluation Algorithm 9.4 for L-MDatalog with $L \in \mathcal{UMD}$ by imposing a limit l on the lengths of modalities that can occur in the evaluation.

Definition 9.3 Let $l \geq 1$ be a fixed number. We define the operator l-$rSatNF_L(J)$ in the same way as $rSatNF_L(J)$, except that only derivations consisting of atoms with modal depths not greater than l are accepted. Let Algorithm l-9.4 be the modification of Algorithm 9.4 where $rSatNF_L$ and Sat_L are replaced respectively by l-$rSatNF_L$ and l-Sat_L (for Procedure 9.5).

It is easy to see that soundness (Theorem 9.2), tightness (Theorem 9.5), and PTIME data complexity (Theorem 9.7) hold for the top-down evaluation Algorithm l-9.4 for $L \in \mathcal{UMD}$ when l is fixed. For completeness, we have the following theorem, which can be proved similarly as Theorem 9.4.

Theorem 9.8 *Let $(P, q(x_1, \ldots, x_k))$ be an L-MDatalog query over a schema S, where $L \in \mathcal{UMD}$, and I be an edb instance over $edb(S)$ in L. For every correct answer θ in L of $P \cup P_I \cup \{\leftarrow q(x_1, \ldots, x_k)\}$, there exists a constant l such that the execution of Algorithm l-9.4 for the query $(P, q(x_1, \ldots, x_k))$ on I returns a relation containing the tuple $(x_1, \ldots, x_k)\theta$.*

Therefore, Algorithm l-9.4 really approximates top-down evaluation of L-MDatalog for $L \in \mathcal{UMD}$. Furthermore, it computes approximations with PTIME data complexity.

10 Magic-Set Transformation for MDatalog

The magic-set technique for Datalog simulates the query-subquery evaluation by rewriting a given query to another equivalent one that when evaluated using a bottom-up technique (e.g. the seminaive evaluation) produces only facts produced by the top-down query-subquery evaluation. Adornments are used as in the query-subquery evaluation. To simulate annotations, the magic-set transformation is augmented with subgoal rectification (see, e.g., [1]). Algorithm 9.1 when applied for Datalog looks like a bottom-up evaluation of a certain program containing additional predicates of *input_* relations and supplementary relations. Those relations are "magic sets" for the transformation.

The query-subquery evaluation of Datalog (see, e.g., [1]) uses adornments to simulate SLD-resolution in pushing constant symbols from goals to subgoals. The annotated version of that evaluation uses also annotations to simulate SLD-resolution in pushing repeats of variables from goals to subgoals. With both adornments and annotations, the annotated query-subquery evaluation method of Datalog is represented so that *input_* relations and supplementary relations (as sup_i in Procedure 9.3) consist of tuples of constant symbols, and classical relational operators can be used for them (see, e.g., [1]). For MDatalog, essential information of a goal $\leftarrow \triangle E$ is not only constant symbols occurring in E and repeats of variables in E but also the modal context $\triangle$, which may contain variables and atom variables. That is why in Procedures 9.2 and 9.3 for MDatalog we use *input_* relations as relations of atoms, and sup_i relations as relations of substitutions. We see no way to avoid variables and atom variables in *input_* relations and supplementary relations for top-down evaluation of MDatalog.

How can we simulate top-down evaluation of MDatalog by magic-set technique? There are two problems. The first one is that our top-down evaluation of MDatalog uses operations on sets of tuples that may contain variables and atom variables, while a usual bottom-up evaluation for MDatalog like the seminaive evaluation operates only on relations without variables and atom variables. The second problem is that it is very unnatural to simulate "pushing modal contexts from goals to subgoals" by a bottom-up method. For example, if $\triangle$ is an L-instance of $\triangle'$ using the empty substitution, then to solve $\leftarrow \triangle E$ we can try to solve $\leftarrow \triangle' E$. Thus, $\triangle$ is strengthened to $\triangle'$. On the other hand, when computing direct consequences in fixpoint semantics, if we have a ground atom $\triangle E$, and $\triangle''$ is an L-instance of $\triangle$, then we can use $\triangle'' E$; that is, $\triangle$ is weakened to $\triangle''$. Due to this reverse, it is not easy to simulate "pushing modal contexts from goals to subgoals" by a bottom-up method. One can still do a strict simulation by making severe modifications for the definitions, but it is not worth to do so.

In this section, we extend the magic-set transformation of Datalog for L-MDatalog, where $L \in \mathcal{BMD} \cup \mathcal{UMD}$. The extension does not strictly simulate "pushing modal contexts from goals to subgoals". In some cases, e.g. when $L \in \{KDI4_s5, KDI45, KD4I_g5_a\}$, it completely loosens modal contexts. In other cases, e.g. when $L \in \{KD4_s5_s, KD45_{(m)}\}$, shapes of modal contexts are pushed from goals to subgoals, but some loosening still happens. On the other hand, in contrast with our top-down evaluation of MDatalog, our magic-set technique for MDatalog operates only on relations of tuples without variables and atom variables. Our presentation is without subgoal rectification, but the extension with subgoal rectification is straightforward.

10.1 The Magic-Set Transformation

To illustrate the magic-set transformation, we use the L-MDatalog query $(MRSG, query(y))$, where $MRSG$ is the following extension of the Datalog program RSG [1]:

$$\begin{array}{l}
\Box_3(rsg(x,y) \leftarrow flat(x,y)) \\
\Box_3(rsg(x,y) \leftarrow up(x,x_1), rsg(y_1,x_1), down(y_1,y)) \\
\Box_2(rsg(x,y) \leftarrow \Box_3 up(x,x_1), \Box_2 rsg(y_1,x_1), \Box_1 down(y_1,y)) \\
\Box_1(\Diamond_1 rsg(x,y) \leftarrow up(x,x_1), \Diamond_1 rsg(y_1,x_1), down(y_1,y)) \\
query(y) \leftarrow \Diamond_2 rsg(a,y)
\end{array}$$

and $flat$, up, $down$ are edb predicates. For this example, L can be understood as a modal logic of multi-degree belief, but it is not necessary so.

We first consider *adorned* versions of programs and queries. When being resolved with $MRSG$, the goal $\leftarrow query(y)$ is first replaced by $\leftarrow \Diamond_2 rsg(a,y)$. As the first coordinate of the goal atom is *bound* and the second coordinate is *free*, we denote the new goal by $\leftarrow \Diamond_2 rsg^{bf}(a,y)$, where the superscript 'bf' is called an *adornment*. Now suppose that we want to resolve this goal with the third clause of $MRSG$. To make benefits from adornments, we create an adorned version of the third clause of $MRSG$ and resolve the goal with it. In that adorned clause, the atom in the head should be $rsg^{bf}(x,y)$, and because x is bound and up is an edb predicate, x_1 will be bound. That adorned clause is thus

$$\Box_2(rsg^{bf}(x,y) \leftarrow \Box_3 up(x,x_1), \Box_2 rsg^{fb}(y_1,x_1), \Box_1 down(y_1,y)).$$

Note that we do not write adornments for edb predicates.

The relevant adorned clauses for the query $(MRSG, query(y))$ are as follows:

1. $\Box_3(rsg^{bf}(x,y) \leftarrow flat(x,y))$
2. $\Box_3(rsg^{bf}(x,y) \leftarrow up(x,x_1), rsg^{fb}(y_1,x_1), down(y_1,y))$
3. $\Box_2(rsg^{bf}(x,y) \leftarrow \Box_3 up(x,x_1), \Box_2 rsg^{fb}(y_1,x_1), \Box_1 down(y_1,y))$
4. $\Box_1(\Diamond_1 rsg^{bf}(x,y) \leftarrow up(x,x_1), \Diamond_1 rsg^{fb}(y_1,x_1), down(y_1,y))$
5. $\Box_3(rsg^{fb}(x,y) \leftarrow flat(x,y))$
6. $\Box_3(rsg^{fb}(x,y) \leftarrow down(y_1,y), rsg^{bf}(y_1,x_1), up(x,x_1))$
7. $\Box_2(rsg^{fb}(x,y) \leftarrow \Box_1 down(y_1,y), \Box_2 rsg^{bf}(y_1,x_1), \Box_3 up(x,x_1))$
8. $\Box_1(\Diamond_1 rsg^{fb}(x,y) \leftarrow down(y_1,y), \Diamond_1 rsg^{bf}(y_1,x_1), up(x,x_1))$
9. $query^f(y) \leftarrow \Diamond_2 rsg^{bf}(a,y)$

Note that in the clauses 6 - 8, the order of atoms in the bodies is changed so that the binding of y in $down$ can be "passed" via y_1 to rsg and via x_1 to up. Denote the above program by $MRSG^{ad}$.

Definition 10.1 Formally, an *adornment* γ for an n-ary predicate p is a sequence of n letters 'b' or 'f', and p *adorned by* γ is denoted by p^γ. For $A = \triangle p(t_1, \ldots, t_n)$, where p is an *idb* predicate, we use A^γ to denote $\triangle p^\gamma(t_1, \ldots, t_n)$ and say that a variable x is *bound* in A^γ if there exists $1 \leq j \leq n$ such that $t_j = x$ and $\gamma(j) =$ 'b', otherwise x is *free* in A^γ. If $A = \triangle p(t_1, \ldots, t_n)$ and p is an *edb* predicate, then A^γ denotes the atom A itself (this means that we do not use adornments for *edb* predicates). Given a clause $\varphi = \boxdot(A \leftarrow B_1, \ldots, B_k)$ and an adornment γ for the predicate in A, the *adorned version* of φ w.r.t. γ is $\boxdot(A^\gamma \leftarrow B_1^{\gamma_1}, \ldots, B_k^{\gamma_k})$, where γ_i is specified as follows: if B_i is of the form $\triangle p(t_1, \ldots, t_n)$ and t_j is a constant symbol or a variable bound in A^γ or occurring in $B_1, \ldots, B_{j-1}$ then $\gamma_i(j) =$ 'b', else $\gamma_i(j) =$ 'f'.

Definition 10.2 For an L-MDatalog query $(P, q(x_1, \ldots, x_k))$, let $\overline{f}$ be the adornment for q consisting of letters 'f' and P^{ad} be the program consisting of all adorned versions of all clauses of P that are related with $q^{\overline{f}}$ (i.e. directly or indirectly defining $q^{\overline{f}}$). We call P^{ad} the *adorned program* corresponding to the query $(P, q(x_1, \ldots, x_k))$.

We proceed by giving a further transformation for P^{ad}. Consider, for example, the 8th clause of $MRSG^{ad}$. As the head of the clause is $\Diamond_1 rsg^{fb}(x, y)$, from the point of view of SLD-resolution, "inputs" for the body are bound by a certain relation $\Diamond_1 input\_rsg^{fb}(y)$. From $\Diamond_1 input\_rsg^{fb}(y)$ and $down(y_1, y)$, we create $sup_2^8(y, y_1)$ as a supplement for the 2nd atom in the body of the 8th clause of $MRSG^{ad}$. The clause is thus transformed to the following:

(s8.1) $\Box_1(sup_2^8(y, y_1) \leftarrow \Diamond_1 input\_rsg^{fb}(y), down(y_1, y))$
(s8.2) $\Box_1(\Diamond_1 rsg^{fb}(x, y) \leftarrow sup_2^8(y, y_1), \Diamond_1 rsg^{bf}(y_1, x_1), up(x, x_1))$

Furthermore, $sup_2^8(y, y_1)$ triggers an additional search for $\Diamond_1 rsg^{bf}(y_1, x_1)$, which in turn will trigger a search for $\Box_1 rsg^{bf}(y_1, x_1)$ (due to the axiom (D)). Thus, $sup_2^8(y, y_1)$ adds additional "inputs" for the search for $\Box_1 rsg^{bf}(y_1, x_1)$. Hence, we also have the following clause:

(i8.1) $\Box_1(\Box_1 input\_rsg^{bf}(y_1) \leftarrow sup_2^8(y, y_1))$

As another example, the 7th clause of $MRSG^{ad}$ is transformed to

(s7.1) $\Box_2(sup_2^7(y, y_1) \leftarrow input\_rsg^{fb}(y), \Box_1 down(y_1, y))$
(s7.2) $\Box_2(rsg^{fb}(x, y) \leftarrow sup_2^7(y, y_1), \Box_2 rsg^{bf}(y_1, x_1), \Box_3 up(x, x_1))$
(i7.1) $\Box_2(\Box_2 input\_rsg^{bf}(y_1) \leftarrow sup_2^7(y, y_1))$.

The first clause of $MRSG^{ad}$ is transformed to

(s1.1) $\Box_3(rsg^{bf}(x, y) \leftarrow input\_rsg^{bf}(x), flat(x, y))$.

The last clause of $MRSG^{ad}$ is transformed to

(s9.1) $sup_1^9 \leftarrow input\_query^f$
(s9.2) $query^f(y) \leftarrow sup_1^9, \Diamond_2 rsg^{bf}(a, y)$
(i9.1) $\Box_2 input\_rsg^{bf}(a) \leftarrow sup_1^9$

To trigger a search for the query, we use the following rule

(seed) $input\_query^f \leftarrow$

We now give a formal definition of the magic-set transformation. We start with auxiliary notations. For an atom A of the form $\triangle p^\gamma(t_1, \ldots, t_n)$, where $|\triangle| \leq 1$ and $i_1, \ldots, i_k$ are all the indices such that $\gamma(i_j) =$ 'b' for $1 \leq j \leq k$: by $input\_A$ we denote the atom $\triangle input\_p^\gamma(t_{i_1}, \ldots, t_{i_k})$; by $input\_blf\_A$ we denote $\Box_i p^\gamma(t_{i_1}, \ldots, t_{i_k})$ if $\triangle = \Diamond_i$, and $input\_A$ otherwise.[14] For an adorned clause $\varphi_i = \boxdot(A \leftarrow B_1, \ldots, B_k)$ and $1 \leq j \leq k$, let Sup_j^i be the atom of predicate sup_j^i whose arguments are the variables that occur both in $input\_A, B_1, \ldots, B_{j-1}$ and $B_j, \ldots, B_k, A$.

Definition 10.3 Let $(P, q(x_1, \ldots, x_k))$ be an L-MDatalog query and P^{ad} the corresponding adorned program. We construct P^m as follows: At the beginning let P^m contain only the clause $input\_q^{\overline{f}} \leftarrow$, where $\overline{f}$ is the adornment for q consisting of letters 'f'. Then for each clause $\varphi_i = \boxdot(A \leftarrow B_1, \ldots, B_k)$ of P^{ad} :

[14] *blf* stands for "$\Box$-lifting form".

- If no *idb* predicate occurs in $B_1, \ldots, B_k$ then add to P^m the clause

$$\boxdot(A \leftarrow \mathit{input\_A}, B_1, \ldots, B_k) \qquad \text{(s } i\text{.1)}$$

- Otherwise, let $i_1, \ldots, i_h$ be all the indices such that for each $1 \leq j \leq h$, B_{i_j} is an atom of an adorned *idb* predicate. Then add to P^m the following clauses:

$$\begin{array}{ll}
\boxdot(\mathit{Sup}^i_{i_1} \leftarrow \mathit{input\_A}, B_1, \ldots, B_{i_1-1}) & \text{(s } i\text{.1)} \\
\boxdot(\mathit{Sup}^i_{i_j} \leftarrow \mathit{Sup}^i_{i_{j-1}}, B_{i_{j-1}}, \ldots, B_{i_j-1}) \text{ for every } 1 < j \leq h & \text{(s } i.j\text{)} \\
\boxdot(A \leftarrow \mathit{Sup}^i_{i_h}, B_{i_h}, \ldots, B_k) & \text{(s } i.(h+1)\text{)} \\
\boxdot(\mathit{input\_blf\_B}_{i_j} \leftarrow \mathit{Sup}^i_{i_j}) \text{ for every } 1 \leq j \leq h & \text{(i } i.j\text{)}
\end{array}$$

In the last clause given above, we use $\mathit{input\_blf\_B}_{i_j}$ instead of $\mathit{input\_B}_{i_j}$ because in serial modal logics we have that $\boxdot(\Box_i E \to \Diamond_i E)$, hence we should accept $\boxdot(\Diamond_i \mathit{input\_E} \to \Box_i \mathit{input\_E})$.

The L-MDatalog query $(P^m, q^{\overline{f}}(x_1, \ldots, x_k))$ is the result the magic-set transformation for $(P, q(x_1, \ldots, x_k))$.

10.2 Correctness of the Magic-Set Transformation

Let $(P^m, q^{\overline{f}}(x_1, \ldots, x_k))$ be the result of the magic-set transformation for an L-MDatalog query $(P, q(x_1, \ldots, x_k))$. In order to compare $(P^m, q^{\overline{f}}(x_1, \ldots, x_k))$ with $(P, q(x_1, \ldots, x_k))$ and obtain an equivalence we modify the evaluation of $(P^m, q^{\overline{f}}(x_1, \ldots, x_k))$ in two aspects involved with the fixpoint semantics:

- ignoring adornments in definition of the forward labeled form of an atom,
- extending the set of rules specifying the operator Sat_L (for *input_* atoms).

From now on, we assume the modification that if α is an adorned atom of the form $\triangle \Diamond_i p^\gamma(t_1, \ldots, t_n)$ then the *forward labeled form* of α is $\triangle \langle p(t_1, \ldots, t_n) \rangle_i p^\gamma(t_1, \ldots, t_n)$ instead of $\triangle \langle p^\gamma(t_1, \ldots, t_n) \rangle_i p^\gamma(t_1, \ldots, t_n)$. Under this assumption, we have the following property:

Lemma 10.1 *Let $(P, q(x_1, \ldots, x_k))$ be an L-MDatalog query over a schema S, where $L \in \mathcal{BMD} \cup \mathcal{UMD}$, and P^{ad} be the corresponding adorned program. Let I be an edb instance over edb(S) in L, p a predicate of idb(S), and γ an adornment for p such that p^γ is defined by P^{ad}. Then $P^{ad}_L(I)(p^\gamma) = P_L(I)(p)$.*

Proof. It is straightforward to prove by induction on k that $T_{L,P^{ad},I} \uparrow k\,(p^\gamma) = T_{L,P,I} \uparrow k\,(p)$. The assertion of the lemma immediately follows. •

We use *input_E* to denote an atom of a predicate with prefix *input_*. An atom E standing alone or in an atom of the form $\triangle E$ can be an atom of an arbitrary predicate, except that we exclude predicates with the double prefix *input_input_*.

Observe that, if $\triangle E \leftarrow \triangle' E$ is a rule specifying $r\mathit{Sat}_L$ or rNF_L then we should accept also $\triangle \mathit{input\_E} \to \triangle' \mathit{input\_E}$, where $\mathit{input\_E} = \mathit{input\_p}(t_1, \ldots, t_k)$ if $E = p(t_1, \ldots, t_k)$. We give below additional rules for specifying Sat_L to deal with *input_* atoms:

$L \in \{KDI4_s5, KDI45\}$:	$\triangle input\_E \to \Box_m\, input\_E$ if $\lvert\triangle\rvert \geq 1$
$L = KD4I_g5_a$:	$\triangle input\_E \to \Box_k\, input\_E$ if $\lvert\triangle\rvert \geq 1$ where $g(k)$ is the largest group
$L = KD4_s5_s$:	$\triangle\nabla_i\, input\_E \to \Box_i\, input\_E$
$L = KD45_{(m)}$:	$\triangle\nabla_i\triangle'\, input\_E \to \triangle\Box_i\triangle'\, input\_E$ $\triangle\nabla_i\nabla'_i\, input\_E \to \triangle\Box_i\, input\_E$
$L = sCFG$:	if $\Box_i\varphi \to \Box_{j_1}\ldots\Box_{j_k}\varphi$ is an axiom of L : $\triangle\Box_{j_1}\ldots\Box_{j_k}\triangle' input\_E \to \triangle\Box_i\triangle' input\_E$ $\triangle\nabla_i\triangle' input\_E \to \triangle\Box_{j_1}\ldots\Box_{j_k}\triangle' input\_E$

Note that for $L \in \{KDI4_s5, KDI45, KD4I_g5_a\}$, modal contexts of *input_* atoms are completely loosened (to a modality $\boxdot$ such that every atom $\triangle input\_E$ is an L-instance of some atom from $Sat_L(\{\boxdot input\_E\})$). For the remaining logics, some loosening still happens, but certain properties are preserved (and will be passed from goals to subgoals).

The three following auxiliary lemmas can easily be checked for $L \in \mathcal{BMD} \cup \mathcal{UMD}$.

Lemma 10.2 *Let α and β be ground atoms in L-normal labeled form. Suppose that α is an L-instance of β. Then every $\alpha' \in Sat_L(\{\alpha\})$ is an L-instance of some $\beta' \in Sat_L(\{\beta\})$.*

Lemma 10.3 *Let α be a ground atom of the form $\triangle A$, where $\triangle$ is in L-normal labeled form and A is of the form $\Box_i E$, $\Diamond_i E$, or E. Let α' be the forward labeled form of α. Suppose that $\beta \in NF_L(\{\alpha'\})$. Then α is an L-instance of some atom of $Sat_L(\{\beta\})$.*

Lemma 10.4 *Let $\triangle E$ and $\triangle' E$ be ground atoms in L-normal labeled form. Suppose that $\triangle E$ is an L-instance of some atom of $Sat_L(\{\triangle' E\})$. Then $\triangle' input\_E$ is an L-instance of some atom of $Sat_L(\{\triangle input\_E\})$. (Here, the predicate of E cannot start with input_.)*

We now prove that the magic-set transformation is correct.

Lemma 10.5 *Let $(P, q(x_1, \ldots, x_k))$ be an L-MDatalog query over a schema S, where $L \in \mathcal{BMD} \cup \mathcal{UMD}$, P^{ad} be the corresponding adorned program, and $(P^m, q^{\overline{f}}(x_1, \ldots, x_k))$ be the result of the magic-set transformation for $(P, q(x_1, \ldots, x_k))$. Let I be an edb instance over $edb(S)$ in L and p^γ be an adorned version of $p \in idb(S)$. Then every atom $\alpha \in P^m_L(I)$ of p^γ is an L-instance of some atom from $P^{ad}_L(I)$.*

Proof. It is straightforward to prove this lemma by induction on the number of steps needed to derive α for $P^m_L(I)$, using the observation that when transforming P^{ad} to P^m, a clause $\varphi_i = \boxdot(A \leftarrow B_1, \ldots, B_k)$ with $B_{i_1}, \ldots, B_{i_h}$ being atoms of adorned *idb* predicates is broken into
$\boxdot(Sup^i_{i_1} \leftarrow input\_A, B_1, \ldots, B_{i_1-1})$,
$\boxdot(Sup^i_{i_j} \leftarrow Sup^i_{i_{j-1}}, B_{i_{j-1}}, \ldots, B_{i_j-1})$ for every $1 < j \leq h$,
$\boxdot(A \leftarrow Sup^i_{i_h}, B_{i_h}, \ldots, B_k)$,
where *input_A* plays the role of an additional restriction. •

Lemma 10.6 *Let $(P, q(x_1, \ldots, x_k))$ be an L-MDatalog query over a schema S, where $L \in \mathcal{BMD} \cup \mathcal{UMD}$, P^{ad} be the corresponding adorned program, $(P^m, q^{\overline{f}}(x_1, \ldots, x_k))$ be the result of the magic-set transformation for $(P, q(x_1, \ldots, x_k))$, and I be an edb instance over $edb(S)$ in L. Suppose that $\triangle E \in P^{ad}_L(I)$, $\boxdot$ is the universal modality being a $\Box$-lifting form of $\triangle$, and $\boxdot input\_E$ is an L-instance of some atom of $Sat_L(P^m_L(I))$. Then $\triangle E$ is an L-instance of some atom of $P^m_L(I)$.*

Proof. Let n be the smallest number such that $\triangle E \in T_{L,P^{ad},I}\!\uparrow\! n$. We prove the lemma by induction on n. Suppose that the assertion of the lemma holds for all n with a smaller value. Suppose that $\triangle E \in T_{L,P^{ad},I}\uparrow n$ is created by first applying some clause $\varphi_i = \boxdot'(A \leftarrow B_1, \ldots, B_k)$ of P^{ad} to atoms of $Sat_L(I \cup T_{L,P^{ad},I}\uparrow(n-1))$ to create $\triangle'(A'\theta)$, where A' is the forward labeled form of A and θ is the involved substitution, and then normalizing $\triangle'(A'\theta)$. Let $1 \leq i_1 < \ldots < i_h \leq k$ be all the indices such that B_{i_j} for $1 \leq j \leq h$ are atoms of adorned *idb* predicates. Then P^m contains the following clauses:

(a) $\boxdot'(Sup^i_{i_1} \leftarrow input\_A, B_1, \ldots, B_{i_1-1})$,
(b) $\boxdot'(Sup^i_{i_j} \leftarrow Sup^i_{i_{j-1}}, B_{i_{j-1}}, \ldots, B_{i_j-1})$ for every $1 < j \leq h$,
(c) $\boxdot'(A \leftarrow Sup^i_{i_h}, B_{i_h}, \ldots, B_k)$,
(d) $\boxdot'(input\_blf\_B_{i_j} \leftarrow Sup^i_{i_j})$ for every $1 \leq j \leq h$.

Since $\triangle E \in NF_L(\{\triangle'(A'\theta)\})$, by Lemma 10.3, $\triangle'(input\_A\theta)$ is an L-instance of some atom of $Sat_L(\{\triangle input\_E\})$. By Lemma 10.2 and the assumptions of this lemma, it follows that $\triangle'(input\_A\theta)$ is an L-instance of some atom of $Sat_L(P^m_L(I))$. By the clause (a), this implies that $\triangle'(Sup^i_{i_1}\theta)$ is an L-instance of some atom of $P^m_L(I)$. Consequently, by the clauses (d), $\triangle'(input\_blf\_B_{i_1}\theta)$ is an L-instance of some $\alpha \in T_{0L,P}(P^m_L(I))$. We have $NF_L(\{\alpha\}) \subseteq P^m_L(I)$, hence by Lemma 10.3, α is an L-instance of some atom of $Sat_L(P^m_L(I))$. Hence $\triangle'(input\_blf\_B_{i_1}\theta)$ is an L-instance of some atom of $Sat_L(P^m_L(I))$.

We have that $\triangle'(B_{i_1}\theta)$ is an L-instance of some atom of $Sat_L(T_{L,P^{ad},I}\uparrow(n-1))$. Let $\triangle'' B''_{i_1} \in T_{L,P^{ad},I}\uparrow(n-1)$ be the atom such that $\triangle'(B_{i_1}\theta)$ is an L-instance of some atom of $Sat_L(\{\triangle'' B''_{i_1}\})$. Let $\triangle''' E_{i_1} = \triangle'' B''_{i_1}$ and let $\boxdot'''$ be the universal modality being a $\Box$-lifting form of $\triangle'''$. By Lemma 10.2, $\triangle'(B_{i_1}\theta)$ is an L-instance of some atom of $Sat_L(\{\boxdot''' E_{i_1}\})$. By Lemma 10.4, it follows that $\boxdot''' input\_E_{i_1}$ is an L-instance of some atom of $Sat_L(\{\triangle'(input\_B_{i_1}\theta)\})$. Hence, by Lemma 10.2, $\boxdot''' input\_E_{i_1}$ is an L-instance of some atom of $Sat_L(\{\triangle'(input\_blf\_B_{i_1}\theta)\})$, and is thus also an L-instance of some atom of $Sat_L(P^m_L(I))$. Hence, by the inductive assumption, $\triangle'' B''_{i_1}$ is an L-instance of some atom of $P^m_L(I)$. By Lemma 10.2, it follows that $\triangle'(B_{i_1}\theta)$ is an L-instance of some atom of $Sat_L(P^m_L(I))$.

Analogously, it can be shown that, for $1 < j \leq h$, $\triangle'(Sup^i_{i_j}\theta)$ is an L-instance of some atom of $P^m_L(I)$ and $\triangle'(B_{i_j}\theta)$ is an L-instance of some atom of $Sat_L(P^m_L(I))$. Hence, by the clause (c), $\triangle'(A'\theta)$ is an L-instance of some $\triangle^\dagger(A'\theta) \in T_{0L,P^m}(Sat_L(I \cup P^m_L(I)))$.

If $L = KDI45$ and there are some modalities $\triangle^\dagger$ with that property then we take a minimally general one. It can be shown that $\triangle E \in NF_L(\{\triangle'(A'\theta)\})$ is an L-instance of some atom of $NF_L(\{\triangle^\dagger(A'\theta)\})$. This can easily be checked for the case $L \neq KDI45$. For the case $L = KDI45$, the claim holds due to the Sat_L rule: $\triangle\Box_i\alpha \to \triangle\Box_j\alpha$ if $i > j$. Therefore $\triangle E$ is an L-instance of some atom of $P^m_L(I) = NF_L(T_{0L,P^m}(Sat_L(I \cup P^m_L(I))))$. •

Corollary 10.7 *Let $(P, q(x_1, \ldots, x_k))$ be an L-MDatalog query over a schema S, where $L \in \mathcal{BMD} \cup \mathcal{UMD}$, P^{ad} be the corresponding adorned program, $(P^m, q^{\overline{f}}(x_1, \ldots, x_k))$ be the result of the magic-set transformation for $(P, q(x_1, \ldots, x_k))$, and I be an edb instance over edb(S) in L. Then every atom $q^{\overline{f}}(c_1, \ldots, c_k) \in P^{ad}_L(I)$ belongs to $P^m_L(I)$.*

Proof. Suppose that $q^{\overline{f}}(c_1, \ldots, c_k) \in P^{ad}_L(I)$. By the definition of P^m, $input\_q^{\overline{f}} \in P^m_L(I)$. Hence, by the above lemma, $q^{\overline{f}}(c_1, \ldots, c_k) \in P^m_L(I)$. •

The following theorem states that the magic-set transformation for L-MDatalog is correct.

Theorem 10.8 *Let $(P, q(x_1, \ldots, x_k))$ be an L-MDatalog query over a schema S, where $L \in \mathcal{BMD} \cup \mathcal{UMD}$, $(P^m, q^{\overline{f}}(x_1, \ldots, x_k))$ be the result of the magic-set transformation for $(P, q(x_1, \ldots, x_k))$, and I be an edb instance over edb(S) in L. Then $q^{\overline{f}}(c_1, \ldots, c_k) \in P^m_L(I)$ iff $q(c_1, \ldots, c_k) \in P_L(I)$.*

Proof. This theorem immediately follows from Lemmas 10.1, 10.5, and Corollary 10.7. •

11 Conclusions

In this work, we have given formulations for modal deductive databases and defined the modal query language MDatalog, which is as expressive as the general Horn fragment in modal logics. We have defined modal relational algebra L-SPCU and developed evaluation methods for MDatalog. We have proved the following results for the multimodal logics of belief $KDI4_s5$, $KD4_s5_s$, $KD45_{(m)}$, $KDI45$, $KD4I_g5_a$, and the class $sCFG$ of serial context-free grammar logics:

- The data complexity of MDatalog in $KDI4_s5$, $KD4_s5_s$, $KD45_{(m)}$, $KDI45$ is complete in PTIME.
- L-SPCU algebra queries are equivalent (w.r.t. expressiveness) to L-MDatalog queries using nonrecursive L-MDatalog programs.
- The top-down evaluation algorithm for MDatalog is sound, complete, and tight (w.r.t. the corresponding SLD-resolution calculus for L-MProlog). For the logics $KD4I_g5_a$ and $sCFG$, the algorithm works with an approximation method.
- The magic-set transformation for MDatalog is correct.

The magic-set transformation can be combined with the seminaive evaluation to give a more refined bottom-up evaluation method for MDatalog. For $KD4I_g5_a$ and $sCFG$, it also works with an approximation method. Our magic-set transformation for MDatalog does not strictly simulate our top-down evaluation algorithm because modal contexts of goal atoms cannot be pushed from goals to subgoals in a pure way.

Our formulations and methods are highly modular w.r.t. the base modal logic L and the underlying SLD-resolution calculus for L-MProlog. They are appliable for other serial modal logics. For example, in the report [27], we apply them also for all the basic serial monomodal logics.

This work and our previous work [20] are pioneer works on modal deductive databases. This work covers most topics of the theory of deductive databases for modal logics. It establishes a fundamental basis for the subject of modal deductive databases. There remain, of course, some problems deserving investigation, e.g., MDatalog with negation, behaviors of the redundant elimination operator, or efficient representation of *edb* instances.

Our evaluation methods for MDatalog are significantly useful for the computational theory of modal logics. They are an evidence for the usefulness of the direct approach used for modal logic programming. The translation approaches [7, 28] used in modal logic programming are not suitable for modal deductive databases, because they introduce Skolem function symbols and can make clauses not range-restricted.

Because multimodal logics can be used to reason about multi-degree belief (a kind of uncertainty) and epistemic states of agents, we believe that modal deductive databases will have potential applications.

Acknowledgements

I would like to thank the reviewers for many helpful comments and suggestions.

References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* Addison Wesley, 1995.

[2] H. Aldewereld, W. van der Hoek, and J.-J.Ch. Meyer. Rational teams: Logical aspects of multi-agent systems. *Fundamenta Informaticae*, 63(2–3):159–183, 2004.

[3] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic.* Cambridge University Press, 2002.

[4] M. Cadoli, L. Palopoli, and M. Lenzerini. Datalog and description logics: Expressive power. In S. Cluet and R. Hull, editors, *DBPL-6, LNCS 1369*, pages 281–298. Springer, 1998.

[5] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In I. Horrocks, U. Sattler, and F. Wolter, editors, *Description Logics*, 2005.

[6] M.J. Cresswell and G.E. Hughes. *A New Introduction to Modal Logic.* Routledge, 1996.

[7] F. Debart, P. Enjalbert, and M. Lescot. Multimodal logic programming using equational and order-sorted logic. *Theoretical Computer Science*, 105:141–166, 1992.

[8] L. Fariñas del Cerro and M. Penttonen. Grammar logics. *Logique et Analyse*, 121-122:123–134, 1988.

[9] M. Fitting and R.L. Mendelsohn. *First-Order Modal Logic.* Kluwer Academic Publishers, 1999.

[10] E. Franconi and S. Tessaris. Rules and queries with ontologies: A unified logical framework. In H.J. Ohlbach and S. Schaffert, editors, *PPSWR 2004, LNCS 3208*, pages 50–60. Springer, 2004.

[11] J. W. Garson. Quantification in modal logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic: Volume II: Extensions of Classical Logic*, pages 249–307. Reidel, Dordrecht, 1984.

[12] R. Goré and L.A. Nguyen. Clausal tableau systems for multimodal logics of belief. Available at `http://www.mimuw.edu.pl/~nguyen/papers.html`.

[13] G. Gottlob, E. Grädel, and H. Veith. Linear time Datalog and branching time logic. In *Logic-Based Artif. Int.*, pages 443–467. Kluwer Academic Publishers, 2000.

[14] B.N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: combining logic programs with description logic. In *WWW'2003*, pages 48–57. ACM, 2003.

[15] U. Hustadt, B. Motik, and U. Sattler. Data complexity of reasoning in very expressive description logics. In L.P. Kaelbling and A. Saffiotti, editors, *IJCAI*, pages 466–471. Professional Book Center, 2005.

[16] Ch. Koch and S. Scherzinger. Lecture notes on database theory. `http://www-db.cs.uni-sb.de/teaching/dbth0506/slides/dbth-datalog2.pdf`.

[17] A.Y. Levy and M.-Ch. Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, 104(1-2):165–209, 1998.

[18] J.W. Lloyd. *Foundations of Logic Programming, Second Edition.* Springer-Verlag, 1987.

[19] L.A. Nguyen. Constructing the least models for positive modal logic programs. *Fundamenta Informaticae*, 42(1):29–60, 2000.

[20] L.A. Nguyen. The modal query language MDatalog. *Fundamenta Informaticae*, 46(4):315–342, 2001.

[21] L.A. Nguyen. A fixpoint semantics and an SLD-resolution calculus for modal logic programs. *Fundamenta Informaticae*, 55(1):63–100, 2003.

[22] L.A. Nguyen. The modal logic programming system MProlog. In J.J. Alferes and J.A. Leite, editors, *Proceedings of JELIA 2004, LNCS 3229*, pages 266–278. Springer, 2004.

[23] L.A. Nguyen. On modal deductive databases. In J. Eder, H.-M. Haav, A. Kalja, and J. Penjam, editors, *Proceedings of ADBIS 2005, LNCS 3631*, pages 43–57. Springer, 2005.

[24] L.A. Nguyen. The data complexity of MDatalog in basic modal logics. In R. Kralovic and P. Urzyczyn, editors, *Proceedings of MFCS 2006, LNCS 4162*, pages 729–740. Springer-Verlag, 2006.

[25] L.A. Nguyen. Multimodal logic programming. *Theoretical Computer Science*, 360:247–288, 2006.

[26] L.A. Nguyen. Reasoning about epistemic states of agents by modal logic programming. In F. Toni and P. Torroni, editors, *Proceedings of CLIMA VI, LNAI 3900*, pages 37–56. Springer-Verlag, 2006. A revised version is available at `http://www.mimuw.edu.pl/~nguyen/papers.html`.

[27] L.A. Nguyen. Foundations of modal logic programming: The direct approach. Manuscript (served as a technical report), available at `http://www.mimuw.edu.pl/~nguyen/papers.html`, November 2006 (revised March 2007).

[28] A. Nonnengart. How to use modalities and sorts in Prolog. In C. MacNish, D. Pearce, and L.M. Pereira, editors, *Logics in Artificial Intelligence, European Workshop, JELIA '94, York, UK, September 5-8, 1994, Proceedings*, volume 838 of *LNCS*, pages 365–378. Springer, 1994.

[29] H.J. Ohlbach. A resolution calculus for modal logics. In *Proceedings of CADE-88, LNCS310*, pages 500–516. Springer, 1988.

[30] M.A. Orgun and W.W. Wadge. Towards a unified theory of intensional logic programming. *Journal of Logic Programming*, 13(4):413–440, 1992.