Foundations of Modal Logic Programming: The Direct Approach

Linh Anh Nguyen

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

Release 2.2 : 2010-05-03
(Release 1.0 : 2006-11-02)

Contents

1 Introduction **5**
- 1.1 Classical Logic Programming . . . 5
- 1.2 Previous Works on Modal Logic Programming . . . 7
- 1.3 About this Work . . . 9
- 1.4 An Illustrating Example . . . 11

2 Preliminaries **13**
- 2.1 Definitions for Quantified Modal Logics . . . 13
- 2.2 Basic Monomodal Logics . . . 15
- 2.3 A Class of Basic Serial Multimodal Logics . . . 16
- 2.4 Multimodal Logics of Belief . . . 16
- 2.5 Serial Context-Free Grammar Logics . . . 18
- 2.6 Ordering Kripke Models . . . 19
- 2.7 Substitution and Unification . . . 20
- 2.8 Positive Modal Logic Programs . . . 21
- 2.9 Examples of Application of Modal Logic Programming . . . 24

3 A Framework for Modal Logic Programming **29**
- 3.1 Labeled Modal Operators and Notations . . . 30
- 3.2 Model Generators . . . 32
- 3.3 Fixpoint Semantics . . . 34
- 3.4 SLD-Resolution . . . 37
- 3.5 Soundness and Completeness of SLD-Resolution . . . 39
  - 3.5.1 Soundness . . . 40
  - 3.5.2 Completeness . . . 42
  - 3.5.3 Strong Completeness . . . 45
- 3.6 Summary . . . 46

4 Instantiations of the Framework **47**
- 4.1 A Schema for Semantics of $BSMM$-MProlog . . . 47
- 4.2 A Schema for Semantics of $sCFG$-MProlog . . . 53
- 4.3 Programming about Multi-degree Belief . . . 56
  - 4.3.1 Schemata for Semantics of MProlog in $KDI4_s$ and $KDI4$ . . . 57
  - 4.3.2 A Schema for Semantics of $KDI4_s5$-MProlog . . . 59
  - 4.3.3 A Schema for Semantics of $KDI45$-MProlog . . . 64
- 4.4 Programming in MProlog for Multi-agent Systems . . . 68
  - 4.4.1 A Schema for Semantics of $KD4_s5_s$-MProlog . . . 68
  - 4.4.2 A Schema for Semantics of $KD45_{(m)}$-MProlog . . . 70

4.4.3 A Schema for Semantics of $KD4I_g5_a$-MProlog . . . 73
4.5 Semantics of MProlog in Basic Serial Monomodal Logics . . . 78

5 Optimizations 81
5.1 The Standard Form for Resolution Cycles . . . 81
5.2 Optimizing the Framework and Its Instantiations . . . 82
5.3 Semantics of MProlog-$\Box$ . . . 87
5.4 Restrictions and Iterative Deepening Search . . . 88
5.5 Tabulation . . . 89

6 Design and Implementation of the MProlog System 91
6.1 The Design of MProlog . . . 91
6.1.1 Syntax of MProlog Programs . . . 91
6.1.2 Syntax of SLD-Resolution Calculi for MProlog . . . 92
6.1.3 Main Predicates of MProlog . . . 94
6.1.4 Options of MProlog . . . 95
6.1.5 Examples of MProlog Programs . . . 96
6.2 Implementation of MProlog . . . 98
6.2.1 Data Structures and the Parser of MProlog . . . 98
6.2.2 The Interpreter of MProlog . . . 99
6.2.3 Other Modules of MProlog . . . 101

7 MDatalog and Modal Deductive Databases 103
7.1 Preliminaries . . . 103
7.2 Data Complexity . . . 105
7.3 Modal Relational Algebras . . . 106
7.4 An Approximation Method for L-MDatalog in $L \in \mathcal{UMD}$ . . . 108
7.5 Seminaive Evaluation of MDatalog . . . 110
7.6 Top-Down Evaluation of MDatalog . . . 110
7.6.1 Informal Description . . . 111
7.6.2 A Formal Presentation of the Algorithm . . . 113
7.6.3 Properties of the Algorithm . . . 115
7.6.4 Doing It Set-at-a-Time . . . 118
7.6.5 Further Optimizations . . . 120
7.6.6 The Case $L \in \mathcal{UMD}$ . . . 120
7.7 Magic-Set Transformation for MDatalog . . . 121
7.7.1 The Magic-Set Transformation . . . 122
7.7.2 Correctness of the Magic-Set Transformation . . . 124

8 Discussion and Conclusion 129
8.1 Related Works on Modal Logic Programming . . . 129
8.2 Relation between the Approaches . . . 131
8.3 Conclusions . . . 134

A More about Modules of MProlog 143
A.1 Data Structures Used for MProlog . . . 143
A.2 More about the Parser . . . 145
A.3 Other Predicates . . . 146

B Revisions of this Manuscript **147**

Chapter 1

Introduction

1.1 Classical Logic Programming

Classical logic programming is very useful in practice and has been thoroughly studied by many researchers. There are several efficient implementations of the logic programming language Prolog (e.g., SICStus Prolog and SWI-Prolog) and Prolog has become one of the main declarative programming languages, which is especially useful for programming for AI. The most important topics of the theory of classical logic programming are: semantics of positive logic programs, semantics of logic programs with negation, the theory of deductive databases, and semantics of disjunctive logic programs.

There are three standard semantics for positive logic programs: the least model semantics, the fixpoint semantics, and the SLD-resolution calculus (a procedural semantics) (see, e.g., [44]).

Every positive logic program P has the least Herbrand model M_P, which is the least set of ground atoms that are logical consequences of P. This model has the property that for every positive ground formula φ, $P \models \varphi$ iff $M_P \models \varphi$. As an example, consider the following positive logic program:

$$\begin{array}{l} connected(x,y) \leftarrow edge(x,y) \\ connected(x,y) \leftarrow edge(x,z), connected(z,y) \\ edge(a,b) \leftarrow \\ edge(b,c) \leftarrow \end{array}$$

This program has the least Herbrand model

$$\{edge(a,b), edge(b,c), connected(a,b), connected(b,c), connected(a,c)\}.$$

The fixpoint semantics of positive logic programs was first introduced by van Emden and Kowalski [71]. It states that the least Herbrand model M_P of a positive logic program P can be constructed by repeatedly applying the direct consequence operator T_P, which for a set I of ground atoms returns the set of ground atoms that can directly be derived from I using the clauses of P. This operator is monotonic, continuous, and has the least fixpoint $T_P{\uparrow}\omega = \bigcup_{n=0}^{\omega} T_P{\uparrow}n$, which is equal to M_P. For the above example program, the fixpoint is reached after three rounds:

$$\begin{array}{lcl} T_P{\uparrow}0 & = & \emptyset \\ T_P{\uparrow}1 & = & \{edge(a,b), edge(b,c)\} \\ T_P{\uparrow}2 & = & \{edge(a,b), edge(b,c), connected(a,b), connected(b,c)\} \\ T_P{\uparrow}3 & = & \{edge(a,b), edge(b,c), connected(a,b), connected(b,c), connected(a,c)\} \\ T_P{\uparrow}\omega & = & T_P{\uparrow}3 \end{array}$$

SLD-resolution, named by Apt and van Emden in [6], was first described by Kowalski [40] for logic programming. It is a top-down (or goal-directed) procedure for answering queries in positive logic programs. For example, to show that $\{x/a, y/c\}$ is a correct answer for the query $\exists x \exists y\ connected(x, y)$ w.r.t. the given example program, we can use the following SLD-refutation of the goal $\leftarrow connected(x, y)$, which is the negation of the query:

Goals	Input clauses	MGUs
$\leftarrow connected(x, y)$	$connected(x_1, y_1) \leftarrow edge(x_1, z_1), connected(z_1, y_1)$	$\{x/x_1, y/y_1\}$
$\leftarrow edge(x_1, z_1), connected(z_1, y_1)$	$edge(a, b) \leftarrow$	$\{x_1/a, z_1/b\}$
$\leftarrow connected(b, y_1)$	$connected(x_3, y_3) \leftarrow edge(x_3, y_3)$	$\{x_3/b, y_1/y_3\}$
$\leftarrow edge(b, y_3)$	$\leftarrow edge(b, c)$	$\{y_3/c\}$
the empty clause		

Note that standardizing variables apart is used for input clauses. The composition of the used mgu's when restricted to $\{x, y\}$ gives the computed answer $\{x/a, y/c\}$ of the refutation.

SLD-resolution is sound in the sense that, given a positive logic program P and a goal G, every computed answer θ of $P \cup \{G\}$ is a correct answer of $P \cup \{G\}$ (w.r.t. the least model semantics, i.e. $P \vDash \forall(G\theta)$), and is complete in the sense that for every correct answer θ of $P \cup \{G\}$, there exists a computed answer θ' of $P \cup \{G\}$ such that $G\theta = G\theta'\delta$ for some substitution δ.

SLD-resolution closely relates with the fixpoint semantics. The first one is a top-down procedure, while the second one is a bottom-up procedure for answering queries. The completeness proof of SLD-resolution is based on the computation of $T_{L,P}\uparrow\omega$ and a lifting technique.

A logic program containing negation in bodies of some program clauses is called a normal logic program. There are several declarative semantics for normal logic programs, e.g. the stable-model semantics [33] and well-founded semantics [72], which are not equivalent (in two-valued logic). As procedural semantics, SLDNF-resolution, which uses negation as failure, is the most well known (see, e.g., [21, 41]). In this work, we will not study modal logic programming with negation, so we do not discuss logic programming with negation in details.

Deductive databases are very useful for practical applications. A deductive database consists of an instance of extensional relations and a logic program defining intensional relations using the extensional relations. For the basic case, the used logic program is a Datalog program, which is a positive logic program without function symbols that satisfies the allowedness (range-restrictedness) condition. The condition requires that every variable occurring in the head of a program clause occurs also in the body of the clause. For the case with negation in bodies of program clauses, there are the standard semantics for stratified logic program and various semantics of normal logic programs adopted for Datalog$^\neg$ programs (see, e.g., [1]). In the recent years, the stable-model semantics has widely been used in answer set programming (see, e.g., [4]), which closely relates to the field of deductive databases.

There are well-developed techniques for computing queries in deductive databases. For Datalog queries, they are the top-down query-subquery evaluation algorithm and the bottom-up evaluation method based on the magic-set transformation and the improved seminaive evaluation (see, e.g., [1]). The query-subquery evaluation algorithm for Datalog queries simulates SLD-resolution but does it set-at-a-time and manages to find all answers effectively. The magic-set transformation for Datalog queries simulates the query-subquery evaluation by rewriting a given query to another equivalent one that when evaluated using a bottom-up techniques (e.g. the seminaive evaluation) produces only facts produced by the top-down query-subquery evaluation.

Disjunctive logic programs allow disjunctions of atoms in heads of program clauses. We refer the reader to [47, 46, 45, 50, 13, 51, 64] for disjunctive logic programming and disjunctive

deductive databases. For a further reading on classical logic programming, we recommend Lloyd's book [44].

1.2 Previous Works on Modal Logic Programming

Modal and temporal logics are useful in many areas of computer science. For example, multimodal logics are used in knowledge representation and multi-agent systems by interpreting $\Box_i\varphi$ as "agent i knows/believes that φ is true". Many authors have proposed modal and temporal extensions for logic programming (see [68, 29] for surveys[1]). There are two approaches to modal logic programming: the direct approach [26, 8, 12, 55, 63] and the translation approach [2, 23, 66]. The first approach directly uses modalities, while the second one translates modal logic programs to classical logic programs.

In [23], Debart et al. applied a functional translation technique for logic programs in multimodal logics which have a finite number of modal operators $\Box_i$ and $\Diamond_i$ of any type among KD, KT, $KD4$, $KT4$, KF and interaction axioms of the form $\Box_i\varphi \rightarrow \Box_j\varphi$. The technique is similar to the one used in Ohlbach's resolution calculus for modal logics [67]. Extra parameters are added to predicate symbols to represent paths in the Kripke model, and special unification algorithms are used to deal with them.

In [66], Nonnengart proposed a semi-functional translation (which translates existential modal operators using functional translation and translates universal modal operators using relational translation). His approach uses accessibility relations for translated programs, but with optimized clauses for representing properties of the accessibility relations, and does not modify unification. Nonnengart [66] applied the approach for modal logic programs in all of the basic serial monomodal logics KD, T, KDB, B, $KD4$, $S4$, $KD5$, $KD45$, and $S5$. He also gave an example in a multimodal logic of type $KD45$.

The translation approach is attractive: just translate and it is done. However, the problem is more complicated. Modal logics add more nondeterminism to the search process, which cannot be eliminated but must be dealt with in some way. In the functional translation [23], the modified unification algorithm may return different mgu's, which cause branching. In the semi-functional translation [66], additional nondeterminism is caused by clauses representing frame restrictions of the used modal logic. In the direct approach considered shortly, additional nondeterminism is caused by modal rules which are used as meta clauses. In our opinion, the direct approach is worth to study, as it is one of the main approaches to deal with modalities and may result in a deeper analysis of the problem.

Using the direct approach for modal logic programming, Balbiani et al. [8] gave a declarative semantics and an SLD-resolution calculus for a class of logic programs in the monomodal logics KD, T, and $S4$. The work assumes that the modal operator $\Box$ does not occur in bodies of program clauses and goals. In [12], Baldoni et al. gave a framework for developing declarative and operational semantics for logic programs in multimodal logics which have axioms of the form $[t_1]\ldots[t_n]\varphi \rightarrow [s_1]\ldots[s_m]\varphi$, where $[t_i]$ and $[s_j]$ are universal modal operators indexed by terms t_i and s_j, respectively. In that work, existential modal operators are disallowed in programs and goals.

As far as we know, amongst the works by other authors that use the direct approach for modal logic programming, only the Molog system proposed by Fariñas del Cerro [26] has been implemented. With Molog, the user can fix a modal logic and define or choose the rules to deal with modal operators. Molog can be viewed as a framework which can be instantiated with

[1] The works [66, 12, 55, 63] on modal logic programming are not covered by the surveys.

particular modal logics. The current version of this system has, however, some drawbacks in design. It only says yes or no without giving computed answers. Molog uses a special predicate, named *prolog*, to call formulas of Prolog, which is undesirable when the amount of Prolog code is not small. Molog uses <-- instead of :-, and & instead of ',', which means that Prolog program files cannot be included in Molog programs. As an extension of Molog, the *Toulouse Inference Machine* (TIM) [9] (together with an abstract machine model called TARSKI for implementation [10]) makes it possible for a user to select clauses which cannot exactly unify with the current goal, but just resemble it in some way.

Despite that modal logic programming has been studied in the last two decades by several researchers, it has not received much attention in practice. In general, modal logics have not been widely used in practice either. The problem is probably that, for the existing practical applications, modal logics are not as competitive as other instruments. On the other hand, for complicated problems that require reasoning about belief and common belief like the wise men puzzle, modal logics are a nice and useful instrument. In our opinion, modal logics and modal logic programming may become useful for the emerging field of multi-agent systems. We believe that modal logic programming is one of the most promising areas for application of modal logics.

It is desirable to study also modal extensions of deductive databases. For example, if we treat belief as a kind of uncertainty, then modal deductive databases using multi-degree belief have potential applications. The term "modal deductive databases" is hard to find in the literature of computer science, while the field of temporal deductive databases has received a lot of attentions from researchers (see, e.g., the survey [19]). As far as we know, there are no works by other authors that are devoted to modal deductive databases and formulated using modal logics. The modal Datalog defined in [35, 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., [16, 43, 36, 31, 37, 17, 61]). 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".

In [54], 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 allowedness. It was shown in [54] that MDatalog and eMDatalog have the same expressiveness in normal monomodal logics. The evaluation method proposed in [54] 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 [62], 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.

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 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 [67, 23] or the semi-functional translation [66] is used, then function symbols may appear. That is, translation techniques do not work for modal queries. Also note that the method of systematically check-

ing all ground substitutions is very insufficient and unacceptable. Therefore the study of the computational methods of deductive databases for modal queries is fully justifiable.

1.3 About this Work

In this work, we define a modal logic programming language called MProlog, which is as expressive as the general Horn fragment in modal logics. We give a framework for developing the least model semantics, fixpoint semantics, and SLD-resolution calculi for MProlog programs in modal logics whose frame restrictions consist of the conditions of seriality (i.e. $\forall x\ \exists y\ R_i(x, y)$ for every modal index i) and some classical first-order Horn formulas. Our approach is direct (i.e. not based on translation to classical logic) and in comparison with the works [8, 12] by other authors that also use the direct approach, we do not assume any special restriction on occurrences of $\Box_i$ and $\Diamond_i$ (programs and goals are of a normal form but the language is as expressive as the general modal Horn fragment). Furthermore, our semantics for MProlog programs are formulated closely to the style of classical logic programming (as in Lloyd's book [44]). We prove that under certain expected properties of a concrete instantiation of the framework for a specific modal logic, the SLD-resolution calculus is sound and complete. Our framework for developing semantics for MProlog programs is designed to be modular in the sense that it can be instantiated for different modal logics with a few details and proofs.

We apply our framework for the following modal logics:

- the class $BSMM$ of basic serial multimodal logics, which are parameterized by an arbitrary combination of generalized versions of axioms (T), (B), (4), (5) (in the form, e.g., $4 : \Box_i\varphi \to \Box_j\Box_k\varphi$) and $I : \Box_i\varphi \to \Box_j\varphi$,
- the class $sCFG$ of serial context-free grammar logics, which are serial multimodal logics characterized by axioms of the form $\Box_i\varphi \to \Box_{j_1} \ldots \Box_{j_h}\varphi$,
- the following multimodal logics of belief:
  - $KDI4_s$, $KDI4$, $KDI4_s5$, $KDI45$: for reasoning about multi-degree belief,
  - $KD4_s5_s$: for use in distributed systems of belief,
  - $KD45_{(m)}$, $KD4I_g5_a$: for reasoning about epistemic states of agents, where the second logic can be used for reasoning about common belief of agents,
- the basic serial monomodal logics KD, T, KDB, KB, $KD4$, $S4$, $KD5$, $KD45$, and $S5$.

Although the listed multimodal logics of belief and the basic serial monomodal logics belong to the $BSMM$ class and some of them belong also to the $sCFG$ class, the special SLD-resolution calculi proposed for them are more efficient.

(Note that, in multimodal logic programming, the work [12] by Baldoni et al. considers only so called inclusion multimodal logics and programs without existential modal operators; and using the translation approach, the work [23] by Debart et al. does not consider symmetric multimodal logics, i.e. the ones with the axiom (B) or (5).)

We prove that our instantiations of the framework for the above listed modal logics are correct, i.e. the fixpoint semantics coincides with the least model semantics, and the SLD-resolution calculus is sound and complete. The framework and these results were published partially in our papers [55, 60, 65, 63].

As reported in [56, 57], we have implemented the MProlog system [58], using our framework for modal logic programming as theoretical foundations. In this work, we present the design

and describe the implementation of that system. Our system is written in Prolog as its module. Codes, libraries, and most features of Prolog can be used in MProlog programs in a pure way. The MProlog system has been designed so that users can implement and add SLD-resolution calculi to the system in a modular way. We have implemented SLD-resolution calculi for a number of modal logics, including the mentioned multimodal logics of belief and the basic serial monomodal logics.

In this work, we also give formulations for modal deductive databases and extend the modal query language MDatalog for deductive databases in multimodal logics. An MDatalog program is now understood as an MProlog program without function symbols that satisfies the allowedness condition. Basing on the existing techniques of Datalog, we define modal relational algebras and give the seminaive evaluation algorithm, the top-down evaluation algorithm, and the magic-set transformation for MDatalog queries. Our top-down evaluation algorithm for MDatalog is based on our SLD-resolution calculi for MProlog. The seminaive evaluation algorithm and the magic-set transformation for MDatalog queries are bottom-up evaluation techniques that closely relate to the fixpoint semantics of MProlog programs. The results of this work like soundness, completeness, and tightness of the top-down evaluation algorithm or correctness of the magic-set transformation for MDatalog queries are proved for all the mentioned modal logics, except $BSMM$. We also prove that MDatalog has PTIME data complexity in the logics $KDI4_s5$, $KDI45$, $KD4_s5_s$, $KD45_{(m)}$, $KD5$, $KD45$, $S5$, KD, T, KDB, and B. In the remaining modal logics $sCFG$, $KDI4_s$, $KDI4$, $KD4I_g5_a$, $KD4$, and $S4$, the data complexity of MDatalog is PSPACE-hard, and we propose an approximation method for evaluating MDatalog queries in those logics in polynomial time. Some of our results on MDatalog were published in our papers [59, 62].

The rest of this work is organized as follows.

In the next section of this chapter, we give an example illustrating our direct approach for modal logic programming.

In Chapter 2, we give basic definitions for modal logics and definitions involving with substitution and unification, specify the considered modal logics, present an ordering of Kripke models, and define the modal logic programming language MProlog. We prove that the MProlog language is as expressive as the general modal Horn fragment in normal modal logics. We also present some examples demonstrating the usefulness of modal logic programming.

In Chapter 3, we present our framework for developing semantics of MProlog programs. The chapter starts with an introduction of labeled modal operators, their semantics, and notations that are used throughout the work. It then contains our formulations of the three mentioned semantics of MProlog programs. The chapter ends with a section concerning soundness and completeness of SLD-resolution.

Chapter 4 contains our instantiations of the framework for the considered modal logics and proofs of their correctness.

Chapter 5 contains optimizations for the framework and its instantiations.

Chapter 6 contains the design and implementation of our MProlog system. The first section contains syntax of MProlog programs, syntax and requirements for SLD-resolution calculi of MProlog, main built-in predicates and options of MProlog. We also present there formalizations of the wise men puzzle in MProlog. In the second section of that chapter, we describe the implementation of the MProlog system, including data structures, the parser, the interpreter, and the compiler of the system.

Chapter 7 contains our formulations, methods, and results for MDatalog.

Chapter 8 contains a discussion on related works and conclusions.

1.4 An Illustrating Example

To illustrate our approach of defining semantics for modal logic programs, we give here an example. Let the base logic be the simplest serial multimodal logic $KD_{(m)}$ and P be the following program:

$$\begin{array}{l}\varphi_1 = \Diamond_1 p(a) \leftarrow \\ \varphi_2 = \Box_1(\ \Box_2 q(x) \leftarrow p(x)\ ) \\ \varphi_3 = \Box_1(\ \Diamond_2 r(x) \leftarrow p(x), \Box_2 q(x)\ ) \\ \varphi_4 = \Box_1\Box_2(\ s(x) \leftarrow q(x), r(x)\ ) \\ \varphi_5 = \Box_1(\ t(x) \leftarrow \Diamond_2 s(x)\ )\end{array}$$

When building a $KD_{(m)}$-model graph M for P, to realize φ_1 at the actual world τ we connect τ to a world w via the accessibility relation R_1 and add $p(a)$ to w. The edge connecting τ to w is created due to $\Diamond_1 p(a)$, so we can label it by $\langle p(a)\rangle_1$ (a labeled form of $\Diamond_1$). The world w can be identified by τ and the edge from τ and denoted by the sequence $\tau\langle p(a)\rangle_1$. If we denote τ by the empty sequence then $w = \langle p(a)\rangle_1$. Apart from building M, we want to represent the model corresponding to M by a set I of *atoms*. To keep the information that $p(a)$ is true at w, we add the atom $\langle p(a)\rangle_1 p(a)$ to I. To realize φ_2 at τ, $\Box_2 q(x) \leftarrow p(x)$ is added to w, and then $\Box_2 q(a)$ is also added to w. To keep the fact that $\Box_2 q(a)$ belongs to w, we add $\langle p(a)\rangle_1 \Box_2 q(a)$ to I. Note that I contains both $\langle p(a)\rangle_1 p(a)$ and $\langle p(a)\rangle_1 \Box_2 q(a)$. Apply the rule φ_3 to I, then I should contain also $\langle p(a)\rangle_1 \Diamond_2 r(a)$, which is then replaced by $\langle p(a)\rangle_1 \langle r(a)\rangle_2 r(a)$ due to a similar reason as for φ_1. Since I contains both $\langle p(a)\rangle_1 \Box_2 q(a)$ and $\langle p(a)\rangle_1 \langle r(a)\rangle_2 r(a)$, after applying φ_4, I should contain also $\langle p(a)\rangle_1 \langle r(a)\rangle_2 s(a)$. Finally, applying φ_5 to I, we get also $\langle p(a)\rangle_1 t(a)$. In general, instead of building a model graph for P we can build such a set I of atoms, which is called a *model generator*. The set $I_{KD_{(m)},P} = \{\langle p(a)\rangle_1 p(a), \langle p(a)\rangle_1 \Box_2 q(a), \langle p(a)\rangle_1 \langle r(a)\rangle_2 r(a), \langle p(a)\rangle_1 \langle r(a)\rangle_2 s(a), \langle p(a)\rangle_1 t(a)\}$ is the least set of ground atoms which can be derived from P in $KD_{(m)}$ in this way. This set is obtained as the least fixpoint of a certain operator $T_{KD_{(m)},P}$ and is called the *least $KD_{(m)}$-model generator* of P.

Given a model generator I, we can construct the *standard $KD_{(m)}$-model* for it by building a model graph. During the construction, to realize a formula $\langle E\rangle_i \varphi$ at a world w, where E is a ground classical atom, we connect w via the accessibility relation R_i to the world identified by the sequence $w\langle E\rangle_i$ and add φ to that world. We realize a formula $\Box_i \varphi$ at a world w by adding φ to every world reachable from w via R_i. To guarantee the constructed model graph to be the smallest, each new world is connected via each accessibility relation to an empty world at the time of its creation. It can be shown that the standard $KD_{(m)}$-model of $I_{KD_{(m)},P}$ is a least $KD_{(m)}$-model of P.

Now let us give an SLD-refutation of $P \cup \{G\}$ in $KD_{(m)}$ for $G = \leftarrow \Diamond_1 t(x)$. By the content of $I_{KD_{(m)},P}$, the computed answer should be $\{x/a\}$. The SLD-refutation should trace back the process of deriving the atom $\langle p(a)\rangle_1 t(a)$ of $I_{KD_{(m)},P}$ from P. As a $KD_{(m)}$-resolvent of G and φ_5, we derive a new goal $G_1 = \leftarrow \Diamond_1 \Diamond_2 s(x)$. As a $KD_{(m)}$-resolvent of G_1 and φ_4, we derive the goal $G_2 = \leftarrow \Diamond_1 \Diamond_2 (q(x) \wedge r(x))$. This goal is not desired, as it contains a formula but not atoms in its body. To overcome this problem, the (existential) modality $\Diamond_1 \Diamond_2$ should be fixed first, e.g., to become $\langle X\rangle_1 \langle Y\rangle_2$, then the goal G_2 can be rewritten to $\leftarrow \langle X\rangle_1 \langle Y\rangle_2 q(x), \langle X\rangle_1 \langle Y\rangle_2 r(x)$. The labeling should be done in two steps as follows: the goal $G = \leftarrow \Diamond_1 t(x)$ is first replaced by $G' = \leftarrow \langle X\rangle_1 t(x)$, the next goal in the derivation is $G_1 = \leftarrow \langle X\rangle_1 \Diamond_2 s(x)$, which is then replaced by $G'_1 = \leftarrow \langle X\rangle_1 \langle Y\rangle_2 s(x)$, and then $G_2 = \leftarrow \langle X\rangle_1 \langle Y\rangle_2 q(x), \langle X\rangle_1 \langle Y\rangle_2 r(x)$ is derived from G'_1 and φ_4. We can then strengthen G_2 to $G_3 = \leftarrow \langle X\rangle_1 \Box_2 q(x), \langle X\rangle_1 \langle Y\rangle_2 r(x)$.

Goals	Input clauses	MGUs
$G = \leftarrow \Diamond_1 t(x)$		
$G' = \leftarrow \langle X\rangle_1 t(x)$		
$G_1 = \leftarrow \langle X\rangle_1 \Diamond_2 s(x)$	$\Box_1(t(x_1) \leftarrow \Diamond_2 s(x_1))$	$\{x_1/x\}$
$G_1' = \leftarrow \langle X\rangle_1 \langle Y\rangle_2 s(x)$		
$G_2 = \leftarrow \langle X\rangle_1 \langle Y\rangle_2 q(x), \langle X\rangle_1 \langle Y\rangle_2 r(x)$	$\Box_1 \Box_2(s(x_2) \leftarrow q(x_2), r(x_2))$	$\{x_2/x\}$
$G_3 = \leftarrow \langle X\rangle_1 \Box_2 q(x), \langle X\rangle_1 \langle Y\rangle_2 r(x)$		
$G_4 = \leftarrow \langle X\rangle_1 p(x), \langle X\rangle_1 \langle Y\rangle_2 r(x)$	$\Box_1(\Box_2 q(x_4) \leftarrow p(x_4))$	$\{x_4/x\}$
$G_5 = \leftarrow \langle p(a)\rangle_1 \langle Y\rangle_2 r(a)$	$\Diamond_1 p(a) \leftarrow$	$\{x/a, X/p(a)\}$
$G_6 = \leftarrow \langle p(a)\rangle_1 p(a), \langle p(a)\rangle_1 \Box_2 q(a)$	$\Box_1(\Diamond_2 r(x_6) \leftarrow p(x_6), \Box_2 q(x_6))$	$\{x_6/a, Y/r(a)\}$
$G_7 = \leftarrow \langle p(a)\rangle_1 \Box_2 q(a)$	$\Diamond_1 p(a) \leftarrow$	ε
$G_8 = \leftarrow \langle p(a)\rangle_1 p(a)$	$\Box_1(\Box_2 q(x_8) \leftarrow p(x_8))$	$\{x_8/a\}$
the empty clause	$\Diamond_1 p(a) \leftarrow$	ε

Figure 1.1: An illustrating example for SLD-resolution

Resolving G_3 with φ_2, we obtain $G_4 = \leftarrow \langle X\rangle_1 p(x), \langle X\rangle_1 \langle Y\rangle_2 r(x)$. Now resolve G_4 with φ_1. As explained in the construction of $I_{KD_{(m)},P}$, the atom $\Diamond_1 p(a)$ in the head of φ_1 can be treated as $\langle p(a)\rangle_1 p(a)$. Thus, resolving G_4 with φ_1 results in $G_5 = \leftarrow \langle p(a)\rangle_1 \langle Y\rangle_2 r(a)$ and an mgu $\{x/a, X/p(a)\}$. Further steps are given in Figure 1.1.

Chapter 2

Preliminaries

2.1 Definitions for Quantified Modal Logics

A language for quantified (multi)modal 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 a fixed number). If $m = 1$ then we ignore the subscript i and write $\Box$ and $\Diamond$. 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.

Definition 2.1.1 A *term* is defined inductively as follows: a variable is a term; a constant symbol is a term; if f is an n-ary function symbol and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term.

Definition 2.1.2 A *(well-formed modal) formula* is defined inductively as follows:

- If p is an n-ary predicate symbol and $t_1, \ldots, t_n$ are terms, then $p(t_1, \ldots, t_n)$ is a formula, called a *classical atom*.
- If φ and ψ are formulas, then so are $(\neg\varphi)$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$, $(\Box_i\varphi)$, and $(\Diamond_i\psi)$.
- If φ is a formula and x is a variable, then $(\forall x.\varphi)$ and $(\exists x.\varphi)$ are formulas.

We also write $\varphi \equiv \psi$ for $(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$.

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 2.1.3 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 $m = 1$ then we write R for R_1. If $R_i(w, u)$ holds then we say that the world u is accessible from the world w via R_i.

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

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 2.1.4 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 $p^{M,w}$.

Definition 2.1.5 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 2.1.6 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 2.1.7 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* $M, V, w \vDash \zeta$ for a formula ζ is defined as follows:

$M, V, w \vDash p(t_1, \ldots, t_n)$ iff $(t_1^M[V], \ldots, t_n^M[V]) \in p^{M,w}$;
$M, V, w \vDash \neg\varphi$ iff $M, V, w \nvDash \varphi$;
$M, V, w \vDash \varphi \wedge \psi$ iff $M, V, w \vDash \varphi$ and $M, V, w \vDash \psi$;
$M, V, w \vDash \varphi \vee \psi$ iff $M, V, w \vDash \varphi$ or $M, V, w \vDash \psi$;
$M, V, w \vDash \varphi \rightarrow \psi$ iff $M, V, w \nvDash \varphi$ or $M, V, w \vDash \psi$;
$M, V, w \vDash \Box_i\varphi$ iff for all $v \in W$ such that $R_i(w, v)$, $M, V, v \vDash \varphi$;
$M, V, w \vDash \Diamond_i\varphi$ iff for some $v \in W$, $R_i(w, v)$ and $M, V, v \vDash \varphi$;
$M, V, w \vDash \forall x.\varphi$ iff for all $a \in D$, $(M, V', w \vDash \varphi)$, where $V'(x) = a$ and $V'(y) = V(y)$ for $y \neq x$;
$M, V, w \vDash \exists x.\varphi$ iff there exists $a \in D$ such that $M, V', w \vDash \varphi$, where $V'(x) = a$ and $V'(y) = V(y)$ for $y \neq x$.

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 $M \models \Gamma$. Without the actual world one would define that $M \models \Gamma$ if $M, w \models \Gamma$ 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.

A *logic* can be defined by a set of well-formed formulas, a class of admissible interpretations, and a satisfaction relation. The class of admissible interpretations for a modal logic L is often specified by restrictions on Kripke frames. We refer to such restrictions by *L-frame restrictions* and call frames with such properties *L-frames*.

Definition 2.1.8 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 $\Gamma \models_L \varphi$ 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 $\Gamma \models_L \varphi$ reflects "local semantic consequence" due to the inclusion of actual world . Also note that $\Gamma \models_L \varphi$ means $\forall(\Gamma) \rightarrow \forall(\varphi)$ is an L-tautology.

If as the class of admissible interpretations we take the class of all Kripke models (with no restrictions on the accessibility relations) then we obtain the quantified multimodal logic $K_{(m)}$. This logic is axiomatized by the following system:

- axioms for classical predicate logic (without identity)
- the K-axioms: $\Box_i(\varphi \rightarrow \psi) \rightarrow (\Box_i\varphi \rightarrow \Box_i\psi)$
- the Barcan formula axioms: $\forall x.\Box_i\varphi \rightarrow \Box_i\forall x.\varphi$
- the axioms defining $\Diamond_i$: $\Diamond_i\varphi \equiv \neg\Box_i\neg\varphi$
- the modus ponens rule: $\dfrac{\varphi \quad \varphi \rightarrow \psi}{\psi}$
- the generalization rule: $\dfrac{\varphi}{\forall x.\varphi}$
- and the modal generalization rules: $\dfrac{\varphi}{\Box_i\varphi}$

Note that the converse Barcan formula $\Box_i\forall x.\varphi \rightarrow \forall x.\Box_i\varphi$ is a consequence of this axiomatization system. Every logic whose axiomatization is an extension of the system $K_{(m)}$ is called a *normal multimodal logic*.

A modal logic that extends $K_{(m)}$ by some axioms is called a *normal* modal logic. A modal logic is called a *serial* modal logic if it contains the axiom $\Box_i\varphi \rightarrow \Diamond_i\varphi$, for every $1 \leq i \leq m$.

2.2 Basic Monomodal Logics

When $m = 1$, $K_{(m)}$ is denoted by K. Other normal monomodal logics are obtained by adding to K certain axioms. The most popular axioms used for extending K are (D), (T), (B), (4), and (5), which respectively correspond to seriality, reflexiveness, symmetry, transitiveness, and euclideaness of the accessibility relation R.

Axiom	Schema	Corresponding Condition on R
(D)	$\Box\varphi \to \Diamond\varphi$	$\forall u\, \exists v\ R(u,v)$
(T)	$\Box\varphi \to \varphi$	$\forall u\ R(u,u)$
(B)	$\varphi \to \Box\Diamond\varphi$	$\forall u, v\ R(u,v) \to R(v,u)$
(4)	$\Box\varphi \to \Box\Box\varphi$	$\forall u, v, w\ R(u,v) \wedge R(v,w) \to R(u,w)$
(5)	$\Diamond\varphi \to \Box\Diamond\varphi$	$\forall u, v, w\ R(u,v) \wedge R(u,w) \to R(v,w)$

The names of normal monomodal logics often consist of K and the names of the added axioms, e.g. KDB is the logic which extends K with the axioms (D) and (B). The special cases are T, B, $S4$, and $S5$, which stand for KT, KTB, $KT4$, and $KT5$, respectively.

2.3 A Class of Basic Serial Multimodal Logics

A normal multimodal logic can be characterized by axioms extending the system $K_{(m)}$. Consider the class $BSMM$ of basic serial multimodal logics specified as follows. A $BSMM$ logic is a normal multimodal logic parameterized by relations $AD/1$, $AT/1$, $AI/2$, $AB/2$, $A4/3$, $A5/3$ on the set $\{1, \ldots, m\}$, where the numbers on the right are arities and AD is required to be full (i.e. $AD(i)$ holds for every $1 \leq i \leq m$). These relations specify the following axioms:

$\Box_i\varphi \to \Diamond_i\varphi$	if $AD(i)$
$\Box_i\varphi \to \varphi$	if $AT(i)$
$\Box_i\varphi \to \Box_j\varphi$	if $AI(i,j)$
$\varphi \to \Box_i\Diamond_j\varphi$	if $AB(i,j)$
$\Box_i\varphi \to \Box_j\Box_k\varphi$	if $A4(i,j,k)$
$\Diamond_i\varphi \to \Box_j\Diamond_k\varphi$	if $A5(i,j,k)$

It can be shown that the above axioms correspond to the following frame restrictions in the sense that by adding some of the axioms to the system $K_{(m)}$ we obtain an axiomatization system which is sound and complete with respect to the class of admissible interpretations that satisfy the corresponding frame restrictions.

Axiom	Corresponding Condition
$\Box_i\varphi \to \Diamond_i\varphi$	$\forall u\ \exists v\ R_i(u,v)$
$\Box_i\varphi \to \varphi$	$\forall u\ R_i(u,u)$
$\Box_i\varphi \to \Box_j\varphi$	$R_j \subseteq R_i$
$\varphi \to \Box_i\Diamond_j\varphi$	$\forall u, v\ (R_i(u,v) \to R_j(v,u))$
$\Box_i\varphi \to \Box_j\Box_k\varphi$	$\forall u, v, w\ (R_j(u,v) \wedge R_k(v,w) \to R_i(u,w))$
$\Diamond_i\varphi \to \Box_j\Diamond_k\varphi$	$\forall u, v, w\ (R_i(u,v) \wedge R_j(u,w) \to R_k(w,v))$

For a $BSMM$ logic L, we define the set of L-frame restrictions to be the set of the frame restrictions corresponding to the tuples of the relations AD, AT, AI, AB, $A4$, $A5$.

We also use $BSMM$ to denote an arbitrary logic belonging to the $BSMM$ class.

2.4 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:

$$\begin{aligned} KDI4_s &= K_{(m)} + (D) + (I) + (4_s) \\ KDI4 &= K_{(m)} + (D) + (I) + (4) \\ KDI4_s5 &= K_{(m)} + (D) + (I) + (4_s) + (5) \\ KDI45 &= K_{(m)} + (D) + (I) + (4) + (5) \end{aligned}$$

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". The axioms (5) are controversial as they are quite strong. For this reason, we consider also $KDI4$ and $KDI4_s$. 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 $\{1, \ldots, n\}$. Suppose that an index $1 \leq i \leq m$ stands for the group of agents whose indices form the set $g(i)$. We can adopt the axioms (D), (4), and additionally $(I_g) : \Box_i\varphi \to \Box_j\varphi$ if $g(i) \supset g(j)$ (i.e. i indicates a group that contains the group identified by j), and $(5_a) : \neg\Box_i\varphi \to \Box_i\neg\Box_i\varphi$ if $g(i)$ is a singleton (i.e. 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. [3]. Our modal operator of

common belief satisfies positive introspection, while the operator of mutual belief introduced in [3] 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)	$\forall u,v,w\ (R_i(u,v) \wedge R_i(v,w) \rightarrow R_i(u,w))$
(4_s)	$\forall u,v,w\ (R_j(u,v) \wedge R_i(v,w) \rightarrow R_i(u,w))$
(5)	$\forall u,v,w\ (R_i(u,v) \wedge R_i(u,w) \rightarrow R_i(w,v))$
(5_s)	$\forall u,v,w\ (R_j(u,v) \wedge R_i(u,w) \rightarrow R_i(v,w))$
(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 $1 \leq i \leq m$.

2.5 Serial Context-Free Grammar Logics

A *grammar logic* is a multimodal logic extending $K_{(m)}$ with "inclusion axioms" of the form $\Box_{i_1} \ldots \Box_{i_k}\varphi \rightarrow \Box_{j_1} \ldots \Box_{j_h}\varphi$. Such logics were introduced by Fariñas del Cerro and Penttonen in [28]. An inclusion axiom $\Box_{i_1} \ldots \Box_{i_k}\varphi \rightarrow \Box_{j_1} \ldots \Box_{j_h}\varphi$ 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 where the corresponding side stands for the identity relation if $k = 0$ or $h = 0$.

An inclusion axiom $\Box_{i_1} \ldots \Box_{i_k}\varphi \rightarrow \Box_{j_1} \ldots \Box_{j_h}\varphi$ can also be seen as the grammar rule $i_1 \ldots i_k \rightarrow j_1 \ldots j_h$ 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 $i \rightarrow j_1 \ldots j_k$.

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 $(D) : \Box_i\varphi \rightarrow \Diamond_i\varphi$ (for every $1 \leq i \leq m$).

Proposition 2.5.1 *Let L be an sCFG logic. Then the following conditions are equivalent:*

1. *$\Box_{i_1} \ldots \Box_{i_k}\varphi \rightarrow \Box_{j_1} \ldots \Box_{j_h}\varphi$ is L-valid for any φ.*
2. *$\Box_{i_1} \ldots \Box_{i_k}\varphi \rightarrow \Box_{j_1} \ldots \Box_{j_h}\varphi$ 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 $\mathcal{G}(L)$.*

Proof. The implication (3) $\Rightarrow$ (2) follows by induction on the length of the derivation of $j_1 \ldots j_h$ from $i_1 \ldots i_k$ by the grammar $\mathcal{G}(L)$, using substitution, the K-axiom $\Box_i(\psi \rightarrow \xi) \rightarrow (\Box_i\psi \rightarrow \Box_i\xi)$ and the modal necessitation rule $\psi/\Box_i\psi$.

The implication (2) $\Rightarrow$ (1) is clear.

By the correspondence theory, (1) is equivalent with $(1')$: $R_{j_1} \circ \cdots \circ R_{j_h} \subseteq R_{i_1} \circ \cdots \circ R_{i_k}$ *is a consequence of the L-frame restrictions*, and (2) is equivalent with $(2')$: $R_{j_1} \circ \cdots \circ R_{j_h} \subseteq R_{i_1} \circ \cdots \circ R_{i_k}$ *is a consequence of the L-frame restrictions without using the seriality conditions.*

We show that $(1') \Rightarrow (2')$. Suppose that $(1')$ holds and $(w_0, w_h) \in R_{j_1} \circ \cdots \circ R_{j_h}$. Let $w_1, \ldots, w_{h-1}$ be elements s.t. $w_0 R_{j_1} w_1 \cdots w_{h-1} R_{j_h} w_h$. Let P be the classical positive program specified as follows:

- If $R_{t_1} \circ \ldots \circ R_{t_n} \subseteq R_s$ is an L-frame restriction,
  then $R_s(x_0, x_n) \leftarrow R_{t_1}(x_0, x_1), \ldots, R_{t_n}(x_{n-1}, x_n)$ is a clause of P.
- For every $1 \leq s \leq m$, $R_s(x, f_s(x))$ is a clause of P, where f_s is a Skolem function symbol. (This represents seriality of R_s.)
- P contains unary clauses $R_{j_1}(w_0, w_1), \ldots, R_{j_h}(w_{h-1}, w_h)$, where $w_0, \ldots, w_h$ are constants.

By the assumption $(1')$, there exists a classical SLD-refutation of $P \cup \{\leftarrow\ R_{i_1}(w_0, x_1),\ R_{i_2}(x_1, x_2),\ \ldots,\ R_{i_{k-1}}(x_{k-2}, x_{k-1}),\ R_{i_k}(x_{k-1}, w_h)\}$. Observe that Skolem function symbols f_s cannot occur in this refutation, because once a Skolem function symbol appears, it cannot disappear later. Thus $\exists x_1 \ldots \exists x_{k-1}(R_{i_1}(w_0, x_1),\ R_{i_2}(x_1, x_2),\ \ldots,\ R_{i_{k-1}}(x_{k-2}, x_{k-1}),\ R_{i_k}(x_{k-1}, w_h))$ is a logical consequence of the program obtained from P by excluding the clauses of seriality, and this is independent from values of $w_1, \ldots, w_{h-1}$. Hence $(w_0, w_h) \in R_{i_1} \circ \cdots \circ R_{i_k}$ is a logical consequence of $(w_0, w_h) \in R_{j_1} \circ \cdots \circ R_{j_h}$ and the L-frame restrictions without the seriality conditions. This shows that $(1') \Rightarrow (2')$.

To complete the proof, we can show that $(2') \Rightarrow (3)$. This implication can be proved by induction on the length of a derivation of $R_{j_1} \circ \cdots \circ R_{j_h} \subseteq R_{i_1} \circ \cdots \circ R_{i_k}$ from the L-frame restrictions without the seriality conditions. •

Corollary 2.5.2 *Let L be an sCFG logic, $\boxdot$ and $\boxdot'$ be universal modalities. Then the problem of checking whether $\boxdot'\varphi \rightarrow \boxdot\varphi$ is L-valid for any φ is decidable.*

Proof. This corollary follows from Proposition 2.5.1 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 [22, 30, 32, 14].

2.6 Ordering Kripke Models

A formula is in *negation normal form* if it does not contain the connective $\rightarrow$ 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 2.6.1 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.

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

Definition 2.6.2 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. $\forall x, x', y \;\; R_i(x, y) \wedge r(x, x') \rightarrow \exists y' \; R'_i(x', y') \wedge r(y, y')$, for all $1 \leq i \leq m$.
3. $\forall x, x', y' \;\; R'_i(x', y') \wedge r(x, x') \rightarrow \exists y \; R_i(x, y) \wedge r(y, y')$, 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 2.6.1 *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. [53]).

2.7 Substitution and Unification

A *substitution* is a finite set $\theta = \{x_1/t_1, \ldots, x_k/t_k\}$, where $x_1, \ldots, x_k$ are different variables, $t_1, \ldots, t_k$ are terms, and $t_i \neq x_i$ for all $1 \leq i \leq k$. By ε we denote the *empty substitution.*

An *expression* is either a term or a formula without quantifiers, and a *simple expression* is either a term or an atom.

Let $\theta = \{x_1/t_1, \ldots, x_k/t_k\}$ be a substitution and φ be an expression. Then $\varphi\theta$, the *instance* of φ by θ, is the expression obtained from φ by simultaneously replacing all occurrences of the variable x_i in φ by the term t_i, for $1 \leq i \leq k$.

Let $\theta = \{x_1/t_1, \ldots, x_k/t_k\}$ and $\delta = \{y_1/s_1, \ldots, y_h/s_h\}$ be substitutions (where $x_1, \ldots, x_k$ are distinct variables, and $y_1, \ldots, y_h$ are also distinct variables). Then the *composition* $\theta\delta$ of θ and δ is the substitution obtained from the sequence $\{x_1/(t_1\delta), \ldots, x_k/(t_k\delta), y_1/s_1, \ldots, y_h/s_h\}$ by deleting any binding $x_i/(t_i\delta)$ for which $x_i = (t_i\delta)$ and deleting any binding y_j/s_j for which $y_j \in \{x_1, \ldots, x_k\}$.

If θ and δ are substitutions such that $\theta\delta = \delta\theta = \varepsilon$, then we call them *renaming substitutions.*

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 $\psi = \varphi\gamma$.

A substitution θ is *more general* than a substitution δ if there exists a substitution γ such that $\delta = \theta\gamma$. Note that according to our definition, θ is more general than itself.

Let Γ be a set of simple expressions. A substitution θ is called a *unifier* for Γ if $\Gamma\theta$ is a singleton. If $\Gamma\theta = \{\varphi\}$ then we say that θ unifies Γ (into φ). A unifier θ for Γ is called a *most general unifier* (mgu) for Γ if θ is more general than every unifier of Γ.

There is an effective algorithm, called the *unification algorithm*, for checking whether a set Γ of simple expressions is unifiable (i.e. has a unifier) and computing an mgu for Γ if Γ is unifiable (see, e.g., [44]).

2.8 Positive Modal Logic Programs

In this section, we present a modal logic programming language called MProlog. This language is as expressive as the general Horn fragment in the considered modal logics. For L being one of the basic monomodal logics with axiom (5) or multimodal logics of belief, we adopt some restrictions on MProlog to obtain L-MProlog. The restrictions do not reduce expressiveness of the language and are acceptable from the practical point of view.

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 2.8.1 A *program clause* is a formula of the form $\boxdot(A \leftarrow B_1, \ldots, B_n)$, where $n \geq 0$ and A, B_1, ..., B_n are formulas of the form E, $\Box_i E$, or $\Diamond_i E$ 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.

Definition 2.8.2 An *MProlog program* is a finite set of program clauses.

Definition 2.8.3 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, ..., 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 $P \models_L Q$ iff $P \cup \{G\}$ is L-unsatisfiable. For the proof of this statement, just note that $G = \forall(\neg(\alpha_1 \wedge \ldots \wedge \alpha_k))$.

In the logic $KD5$, we have a tautology $\triangle\nabla\nabla'\varphi \equiv \nabla\nabla'\varphi$, where ∇ and ∇' denote modal operators. This equivalence was given in [18]. In $KD5$, we also have $\Diamond\nabla\varphi \equiv \Box\nabla\varphi$. Furthermore, in the logics $KD45$ and $S5$, we have $\nabla\nabla'\varphi \equiv \nabla'\varphi$. So, for these logics we refine the definitions of MProlog programs and goals as follows.

Definition 2.8.4 For $L \in \{KD5, KD45, S5\}$, an *L-MProlog program* is a finite set of program clauses of the form $\boxdot(A \leftarrow B_1, \ldots, B_n)$, where the length of $\boxdot$ is not greater than 2 for $L = KD5$, and not greater than 1 for $L \in \{KD45, S5\}$. An MProlog goal is called an *L-MProlog goal* if its modal depth is not greater than 2 for $L = KD5$, and not greater than 1 for $L \in \{KD45, S5\}$. (Recall that the modal depth of φ is the maximal nesting depth of modal operators occurring in φ.)

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 2.8.5 For $L \in \{KDI4_s, KDI4, KDI4_s5, KDI45, KD4_s5_s\}$, 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.

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 2.8.6 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 three above definitions, assume that no restriction is adopted for the form of L-MProlog programs and goals. In the following, we define an extension of MProlog called *eMProlog*, which stands for the general modal Horn fragment.

Definition 2.8.7 A formula φ without quantifiers is called a *non-negative modal Horn formula (without quantifiers)* if one of the following conditions holds:

- φ is a classical atom;
- $\varphi = \psi \leftarrow \zeta$, where ψ is a non-negative modal Horn formula and ζ is a positive formula in negation normal form;
- $\varphi = \Box_i\psi$ or $\varphi = \Diamond_i\psi$ or $\varphi = \psi \wedge \zeta$, where ψ and ζ are non-negative modal Horn formulas.

Definition 2.8.8 An *eMProlog program* is a finite set of formulas of the form $\forall(\varphi)$, where φ is a non-negative modal Horn formula without quantifiers. An *eMProlog query* is a formula of the form $\exists(\varphi)$, where φ is a positive formula without quantifiers. An *eMProlog goal* is the negation of an eMProlog query.

We now define answers and correct answers.

Definition 2.8.9 Let P be an MProlog (resp. eMProlog) program and G an MProlog (resp. eMProlog) goal. An *answer* θ for $P \cup \{G\}$ is a substitution for variables of G (i.e. if $x_1, \ldots, x_n$ are all variables of G, then $\theta = \{x_{i_1}/t_1, \ldots, x_{i_k}/t_k\}$ for some $1 \leq i_1 < \ldots < i_k \leq n$ and some terms $t_1, \ldots, t_k$).

Definition 2.8.10 Let L be a modal logic, P an MProlog (resp. eMProlog) program, $Q = \exists(\varphi)$ an MProlog (resp. eMProlog) query and G the corresponding goal (i.e. $G = \neg Q$). Let θ be an answer for $P \cup \{G\}$. We say that θ is a *correct answer* in L for $P \cup \{G\}$ if $P \vDash_L \forall(\varphi\,\theta)$.

The following proposition states that MProlog and L-MProlog have the same expressiveness as eMProlog.

Proposition 2.8.1 *Let L be any modal logic introduced in this section. For any eMProlog program P and any eMProlog goal G, there exist an MProlog program P' and an MProlog goal G' such that:*

- *Every correct answer in L for $P \cup \{G\}$ is a correct answer in L for $P' \cup \{G'\}$ and vice versa.*
- *If $L \in \{KD5, KD45, S5, KDI4_s, KDI4, KDI4_s5, KDI45, KD4_s5_s, KD45_{(m)}, KD4I_g5_a\}$, then P' is an L-MProlog program and G' is an L-MProlog goal.*

- *P' and G' can be obtained from P and G in polynomial time.*

Proof. For any formula φ with n variables and no quantifiers, by p_φ we denote a fresh n-ary predicate symbol, and by E_φ we denote the classical atom $p_\varphi(x_1,\dots,x_n)$, where $x_1,\dots,x_n$ are the variables occurring in φ. In this proof we use Γ and Λ to denote sequences of formulas.

Let P be an eMProlog program and $G = \leftarrow \xi$ an eMProlog goal. Let $P_2 = P \cup \{E_\xi \leftarrow \xi\}$ and $G' = \leftarrow E_\xi$. We first apply translation rules that replace a complex formula by a fresh atom which is defined by that formula. The method was first used by Mints [52] for modal logics. The translation rules are given in the below list, in which each row contains a formula to be replaced in the left, and the replacing ones in the right. The symbol $\boxdot$ in the rules presents a universal modality. The formula φ in the rules (2.1), (2.3) and (2.4), and ψ in the rules (2.7) and (2.8) are required not to be a classical atom. The rules (2.5) and (2.9) are designed in the way to shorten the result. We apply the translation rules to P_2 until no further changes can be made. Let P_3 be the resulting set.

$$\forall(\boxdot\Diamond_i\varphi) \qquad \forall(\boxdot\Diamond_i E_\varphi),\ \forall(\boxdot\Box_i(\varphi \leftarrow E_\varphi)) \tag{2.1}$$

$$\forall(\boxdot(\varphi \wedge \psi)) \qquad \forall(\boxdot\varphi),\ \forall(\boxdot\psi) \tag{2.2}$$

$$\forall(\boxdot(\Box_i\varphi \leftarrow \Gamma)) \qquad \forall(\boxdot(\Box_i E_\varphi \leftarrow \Gamma)),\ \forall(\boxdot\Box_i(\varphi \leftarrow E_\varphi)) \tag{2.3}$$

$$\forall(\boxdot(\Diamond_i\varphi \leftarrow \Gamma)) \qquad \forall(\boxdot(\Diamond_i E_\varphi \leftarrow \Gamma)),\ \forall(\boxdot\Box_i(\varphi \leftarrow E_\varphi)) \tag{2.4}$$

$$\forall(\boxdot(\varphi \wedge \psi \leftarrow \Gamma)) \qquad \forall(\boxdot(E_{\varphi\wedge\psi} \leftarrow \Gamma)),\ \forall(\boxdot(\varphi \leftarrow E_{\varphi\wedge\psi})),\ \forall(\boxdot(\psi \leftarrow E_{\varphi\wedge\psi})) \tag{2.5}$$

$$\forall(\boxdot((\varphi \leftarrow \psi) \leftarrow \Gamma)) \qquad \forall(\boxdot(\varphi \leftarrow \psi, \Gamma)) \tag{2.6}$$

$$\forall(\boxdot(\varphi \leftarrow \Gamma, \Box_i\psi, \Lambda)) \qquad \forall(\boxdot(\varphi \leftarrow \Gamma, \Box_i E_\psi, \Lambda)),\ \forall(\boxdot\Box_i(E_\psi \leftarrow \psi)) \tag{2.7}$$

$$\forall(\boxdot(\varphi \leftarrow \Gamma, \Diamond_i\psi, \Lambda)) \qquad \forall(\boxdot(\varphi \leftarrow \Gamma, \Diamond_i E_\psi, \Lambda)),\ \forall(\boxdot\Box_i(E_\psi \leftarrow \psi)) \tag{2.8}$$

$$\forall(\boxdot(\varphi \leftarrow \Gamma, \psi \vee \zeta, \Lambda)) \qquad \forall(\boxdot(\varphi \leftarrow \Gamma, E_{\psi\vee\zeta}, \Lambda)), \forall(\boxdot(E_{\psi\vee\zeta} \leftarrow \psi)), \forall(\boxdot(E_{\psi\vee\zeta} \leftarrow \zeta)) \tag{2.9}$$

$$\forall(\boxdot(\varphi \leftarrow \Gamma, \psi \wedge \zeta, \Lambda)) \qquad \forall(\boxdot(\varphi \leftarrow \Gamma, \psi, \zeta, \Lambda)) \tag{2.10}$$

It is easily seen that P_3 is an MProlog program. If L is not one of the modal logics for which there are some restrictions on L-MProlog, then just take $P' = P_3$.

If L is $KD45$ or $S5$, then replace every modal context $\boxdot\Box$ in P_3 by $\Box$ (note that $\boxdot\Box\varphi \equiv \Box\varphi$ is L-valid). If L is $KD5$, then replace every modal context $\boxdot\Box\Box$ in P_3 by $\Box\Box$ (note that $\boxdot\Box\Box\varphi \equiv \Box\Box\varphi$ is L-valid). Let P' be the resulting program.

If L is $KDI4_s5$ or $KD4_s5_s$, then replace every modal context $\boxdot\Box_i$ in P_3 by $\Box_i$ (note that $\boxdot\Box_i\varphi \equiv \Box_i\varphi$ is L-valid), and let P' be the resulting program.

If $L = KD45_{(m)}$, then repeatedly replace every sequence $\Box_i\Box_i$ in modal contexts of program clauses of P_3 by $\Box_i$ (note that $\Box_i\Box_i\varphi \equiv \Box_i\varphi$ is $KD45_{(m)}$-valid), and let P' be the resulting program. If $L = KD4I_g5_a$ then let P' be specified similarly but with the condition that $g(i)$ is singleton.

For $L \in \{KDI4_s, KDI4, KDI45\}$: Repeatedly replace every clause $\boxdot\Box_i(A \leftarrow B_1,\dots,B_k)$ in P_3, where $\boxdot$ is not empty, by $\boxdot E_{\Box_i\varphi}$, $\Box_m(\Box_i E_\varphi \leftarrow E_{\Box_i\varphi})$, and $\Box_m(A \leftarrow B_1,\dots,B_k,E_\varphi)$, where $\varphi = (A \leftarrow B_1,\dots,B_k)$. (Note that $\Box_m\psi \rightarrow \boxdot\psi$ follows from the axioms (4) and (I).) Denote the resulting set by P'.

It is clear that for L being one of the above mentioned modal logics, P' is an L-MProlog program. The whole translation is done in polynomial time.

Suppose that θ is a correct answer in L for $P' \cup \{G'\}$. Let M be an arbitrary L-model of P. We show that $M \vDash \forall(\xi\theta)$. Let M' be an L-model with the same frame as M such that the content of each world w in M' is the least extension of the content of w in M with the property that for any formula φ with E_φ occurring in P', $M', w \vDash \forall(E_\varphi \leftarrow \varphi)$. Thus M' is

an L-model of P' and $M' \vDash \forall(E_\xi \rightarrow \xi)$. Since θ is a correct answer in L for $P' \cup \{G'\}$, we have $M' \vDash \forall(E_\xi\theta)$, and hence $M' \vDash \forall(\xi\theta)$. It follows that $M \vDash \forall(\xi\theta)$ (because M' differs from M only on the semantics of new predicate symbols). Therefore every correct answer in L for $P' \cup \{G'\}$ is a correct answer in L for $P \cup \{G\}$.

For the conversion, suppose that θ is a correct answer in L for $P \cup \{G\}$. Let M' be an L-model of P'. Thus, M' is an L-model of P and $M' \vDash \forall(E_\xi \leftarrow \xi)$. Since θ is a correct answer in L for $P \cup \{G\}$, we have $M' \vDash \forall(\xi\theta)$, and hence $M' \vDash \forall(E_\xi\theta)$. Therefore every correct answer in L for $P \cup \{G\}$ is a correct answer in L for $P' \cup \{G'\}$. •

We will consider also the version MProlog-$\Box$ of MProlog in which existential modal operators are disallowed. As we will see, MProlog-$\Box$ allows much more simpler declarative and procedural semantics than MProlog.

2.9 Examples of Application of Modal Logic Programming

In this section, we present four examples demonstrating the usefulness of modal logic programming. The first example involves reasoning about multi-degree belief, the second one involves distributed systems of belief, the third one formalizes the wise men puzzle, and the last one involves inheritance in a hierarchy of classes. Other examples can be found, e.g., in the work by Baldoni et al. [12].

Example 2.9.1 Assume that there are 5 degrees of belief. Consider the following program P_{mdb}:

$$\varphi_1 = \Box_4 good\_in\_maths(x) \leftarrow maths\_teacher(x)$$
$$\varphi_2 = \Box_5(\Box_i good\_in\_maths(x) \leftarrow \Box_i mathematician(x))$$
$$\varphi_3 = \Box_3(\Diamond_i good\_in\_maths(x) \leftarrow maths\_student(x))$$
$$\varphi_4 = \Box_3(\Diamond_i good\_in\_physics(x) \leftarrow physics\_student(x))$$
$$\varphi_5 = \Box_2(\Diamond_2 good\_in\_maths(x) \leftarrow good\_in\_physics(x))$$
$$\varphi_6 = maths\_teacher(John) \leftarrow$$
$$\varphi_7 = \Box_2 mathematician(Tom) \leftarrow$$
$$\varphi_8 = \Box_5 maths\_student(Peter) \leftarrow$$
$$\varphi_9 = \Box_5 physics\_student(Mike) \leftarrow$$

The index i in the above clauses can take any value from the range 1..5. For the goal $\leftarrow \Box_4 good\_in\_maths(x)$, in all of the logics $KDI4_s$, $KDI4$, $KDI4_s5$, and $KDI45$, we have the correct answer $\{x = John\}$. For the goal $\leftarrow \Box_2 good\_in\_maths(x)$, in the logics $KDI4_s5$ and $KDI45$, we have the additional correct answer $\{x = Tom\}$. For the goal $\leftarrow \Diamond_1 good\_in\_maths(x)$, in all of the mentioned logics, we have three correct answers $\{x = John\}$, $\{x = Tom\}$, and $\{x = Peter\}$. The goal $\leftarrow \Diamond_2 good\_in\_maths(Mike)$ successes only in $KDI4_s5$.

Example 2.9.2 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.

Example 2.9.3 The wise men puzzle is a famous benchmark introduced by McCarthy [48] for AI. It can be stated as follows (cf. [39]). A king wishes to know whether his three advisors (A, B, C) are as wise as they claim to be. Three chairs are lined up, all facing the same direction, with one behind the other. The wise men are instructed to sit down in the order A, B, C. Each of the men can see the backs of the men sitting before them (e.g. C can see A and B). The king informs the wise men that he has three cards, all of which are either black or white, at least one of which is white. He places one card, face up, behind each of the three wise men, explaining that each wise man must determine the color of his own card. Each wise man must announce the color of his own card as soon as he knows what it is. All know that this will happen. The room is silent; then, after a while, wise man A says "My card is white!".

The wise men puzzle has been previously studied in a number of works (e.g., [48, 39, 27, 24, 20, 7, 66, 11]). McCarthy [48] directly used possible worlds to formalize the puzzle. Konolige [39], Fariñas del Cerro and Herzig [27], Nonnengart [66], and Baldoni [11] also used modal logics to formalize the puzzle: Konolige [39] focused on limited reasoning, Fariñas del Cerro and Herzig used the implemented modal logic programming system MOLOG [27], Nonnengart [66] used semi-functional translation for modal logic programming, and Baldoni [11] used a

prefixed tableau system. McCarthy [48], Fariñas del Cerro and Herzig [27], and Nonnengart [66] all used some feature of mutual belief, but they did not define it purely. Baldoni [11] adopted too strong versions of axioms (4) and (5), which are rather not suitable for the puzzle. As other approaches for the wise men puzzle, Elgot-Drapkin [24] used step-logics, while Cimatti and Serafini [20], Attardi and Simi [7] studied reasoning in belief-contexts. Our formalization of the wise men puzzle given below uses $KD4I_g5_a$-MProlog. It is more elegant than the above-mentioned formalizations, as it uses a modal logic with a clear semantics of common belief.

For clarity, instead of numeric indices we use a, b, c, ab, ac, bc, abc with the meaning that $g(a) = \{a\}$, $g(b) = \{b\}$, $g(c) = \{c\}$, ..., and $g(abc) = \{a, b, c\}$. Let P_{wise_men} be the program consisting of the following clauses:

% If Y sits behinds X then X's card is white if Y considers this as possible.
$\varphi_1 = \Box_{abc}\,(white(a) \leftarrow \Diamond_b\, white(a))$
$\varphi_2 = \Box_{abc}\,(white(a) \leftarrow \Diamond_c\, white(a))$
$\varphi_3 = \Box_{abc}\,(white(b) \leftarrow \Diamond_c\, white(b))$
% The following clauses are "dual" to the above ones.
$\varphi_4 = \Box_{abc}\,(\Box_b\, black(a) \leftarrow black(a))$
$\varphi_5 = \Box_{abc}\,(\Box_c\, black(a) \leftarrow black(a))$
$\varphi_6 = \Box_{abc}\,(\Box_c\, black(b) \leftarrow black(b))$
% At least one of the wise men has a white card.
$\varphi_7 = \Box_{abc}\,(white(a) \leftarrow black(b), black(c))$
$\varphi_8 = \Box_{abc}\,(white(b) \leftarrow black(c), black(a))$
$\varphi_9 = \Box_{abc}\,(white(c) \leftarrow black(a), black(b))$
% Each of B and C does not know the color of his own card.
% In particular, each of the men considers that it is possible that his own card is black.
$\varphi_{10} = \Box_{abc}\Diamond_b\, black(b)$
$\varphi_{11} = \Box_{abc}\Diamond_c\, black(c)$

The goal is $\leftarrow \Box_a white(a)$.

Example 2.9.4 In [12], Baldoni et al. formalize an example of inheritance taken from [15] by a modal logic program. We adopt here their example with a small modification. Let us consider four classes: *animal*, *horse*, *bird*, and *tweety*. Since what is true for animals is also true for birds and horses, the *bird* and *horse* classes inherit from the *animal* class. Moreover, the class *tweety* inherits from *bird* and thus from *animal*.

Animals are not agents in the normal sense, but there is a similarity between the mentioned hierarchy of classes with a multi-agent system. We can treat the class *animal* as a group of agents, the class *bird* as its subgroup of agents, and so on. Program clauses with modal context *animal* can be applied for *horse*, *bird*, and *tweety*. Apparently, this feature is useful for defining epistemic states of groups of agents.

In the following, we use the mentioned classes as modal indices and write, e.g., $[animal]$ and $\langle animal\rangle$ respectively for $\Box_{animal}$ and $\Diamond_{animal}$. We use $KD4I_g5_a$, and for the hierarchy, we adopt the conditions that $g(animal) \supset g(horse)$, $g(animal) \supset g(bird)$, and $g(bird) \supset g(tweety)$. Furthermore, treating *tweety* as an object, we assume that $g(tweety)$ is a singleton.

As an example, we have the following clauses:

$$[animal]mode(walk).$$
$$[animal](mode(run) \leftarrow no\_of\_legs(X), X \geq 2).$$
$$[animal](mode(gallop) \leftarrow no\_of\_legs(X), X = 4).$$
$$[horse]no\_of\_legs(4).$$
$$[horse]covering(hair).$$
$$[bird]no\_of\_legs(2).$$
$$[bird]covering(feather).$$
$$[tweety]owner(fred).$$

The atom $[tweety]mode(run)$ can be derived from the above program in $KD4I_g5_a$. If the program contains also $[bird](mode(fly) \leftarrow light)$ or $[bird](mode(fly) \leftarrow \langle bird\rangle light)$, and $\langle tweety\rangle light$, then we can derive $\langle tweety\rangle mode(fly)$ (i.e. have a refutation for the goal $\leftarrow \langle tweety\rangle mode(fly)$). This and the example about the wise men puzzle demonstrate that using $KD4I_g5_a$-MProlog we can reason about possibility. This feature was not incorporated in [12] by Baldoni et al. (as they studied only modal logic programs without existential modal operators).

Chapter 3

A Framework for Modal Logic Programming

As mentioned earlier, there are three standard semantics for classical definite logic programs: the least model semantics, the fixpoint semantics and the SLD-resolution calculus (a procedural semantics). See the works by Lloyd [44] and Apt [5] for foundations of classical logic programming. In this chapter, we give a framework for developing such mentioned semantics for L-MProlog programs. The base logic L is required to be a serial normal modal 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.[1] Consider, for example, the following program in the non-serial modal logic K :

$$\Box p \leftarrow$$
$$q \leftarrow \Diamond p$$
$$s \leftarrow \Box r$$

If there exists a world accessible from the actual world then $\Box p$ implies $\Diamond p$, which then implies q. If there does not then $\Box r$ holds and implies s. The program is thus "nondeterministic" because the accessibility relation is not serial, and consequently, it does not have any least K-model. Apart from the least model semantics, seriality is 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 chapter, we prove the main results using certain lemmas and theorems, which are strongly dependent on L and left as "expected". For a specific logic L, lemmas and theorems with that remark need to be proved to guarantee correctness of the main theorems w.r.t. that logic.

Our framework for developing semantics of MProlog programs is designed to be modular in the sense that it can be instantiated for different modal logics with a few details and proofs. In fact, we are able to specify all the three mentioned semantics for MProlog programs in any of the considered serial modal logics using only one small table that is based on the

[1] In [53], we proved that every positive propositional modal logic program has a least L-model in any serial modal logic $L \in \{KD, T, KDB, B, KD4, S4, KD5, KD45, S5\}$ and can be "characterized" by two minimal L-models if L is one of the almost serial modal logics KB, $K5$, $K45$, $KB5$. On the other hand, there exist positive propositional modal logic programs that cannot be "characterized" by a finitely bounded number of models in the non-serial modal logic K, and there exists a positive propositional modal logic program that cannot be "characterized" by a finite number of models in the non-serial modal logic $K4$ (see [53]).

framework. Furthermore, we need to prove only "expected" lemmas and theorems for a concrete instantiation of the framework, while several important proofs given in this chapter remain unchanged. The "expected" lemmas point out a way for constructing a correct schema for semantics of MProlog. For modularity, proofs of "expected" lemmas and theorems that are strongly dependent on a specific logic are not presented in this chapter but put into a section concerning that logic in the next chapter.

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 $T_{L,P}$, 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 task we label the modal operator $\Diamond_i$. Labeling allows us 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 $\leftarrow \Diamond_i A, \Diamond_i B$. But if we label the operator $\Diamond_i$, let's say by X, to fix it, then we can safely change $\leftarrow \langle X\rangle_i(A \wedge B)$ to $\leftarrow \langle X\rangle_i A, \langle X\rangle_i B$.

We will use the following notations:

- $\top$: the *truth* symbol, with the usual semantics;[2]
- 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 $\langle X\rangle_i$).

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. ∇', $\nabla^{(i)}$, $\nabla^{(i')}$.

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

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

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. The other definitions involving with substitution and unification change accordingly in the usual way.

Definition 3.1.1 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 $EdgeLabels = \{\langle E\rangle_i \mid E \in \mathcal{B} \cup \{\top\}$ 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 3.1.2 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 3.1.3 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 $\langle\top\rangle_i$, which is formalized by the following definition.[3]

Definition 3.1.4 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 $\delta_\top = \{X/\top \mid X$ 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 finite formula set and φ a formula – both without labeled modal operators, $\Gamma \vDash_L \forall(\varphi)$ iff $\Gamma \vDash_L \forall_c(\varphi)$.

[3]Atom variables appear only in *goal* bodies (see Definition 2.8.3). 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 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 3.2.1 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 3.2.2 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.)

As an example, define that a modality is in $KDI4_s5$-normal form if its length is 0 or 1. (This is justified by the $KDI4_s5$-tautology $\nabla\nabla'\varphi \equiv \nabla'\varphi$ with ∇ and ∇' being unlabeled modal operators.) In this example, let $F \neq \top$. Then 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 3.2.3 An *L-normal model generator* is a set of ground atoms in L-normal form and not containing $\Diamond_i$, $\langle\top\rangle_i$, $\top$.

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 (cf. [53]). 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 3.2.4 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 $\alpha \rightarrow \beta$, 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 3.2.5 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.

As an example, for $L = KDI4_s5$, the operator Ext_L is specified by the only rule: $\Box_i E \rightarrow \Box_j E$ if $i > j$; and $Ext_L(\{\Box_2 E\}) = \{\Box_2 E, \Box_1 E\}$.

Definition 3.2.6 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), $\tau = \epsilon$, $H(\tau) = Ext_L(I) \cup Serial_L$. 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 $Serial_L$).[4] 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.

Example 3.2.1 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 $\Box_i E \rightarrow \Box_j E$ 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 $q(b)$.

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

Definition 3.2.7 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 $\langle X\rangle_1\langle p(a)\rangle_1\Diamond_2 q(b)$.

The following lemma will be used to prove, among others, Lemma 3.2.2.

Lemma 3.2.1 *Let I be an L-normal model generator and $M = \langle W, \tau, R_1, \ldots, R_m, H\rangle$ the standard L-model graph of I. Let $w = \langle E_1\rangle_{i_1} \ldots \langle E_k\rangle_{i_k}$ be a world of M and $\triangle = w$ be a modality. Then for α not containing $\top$, $\alpha \in H(w)$ iff there exists a $\Box$-lifting form $\triangle'$ of $\triangle$ such that $\triangle'\alpha \in Ext_L(I)$.*

This lemma can be proved by induction on the length of w in a straightforward way.

The expected results concerning model generators are:

Expected Lemma 3.2.2 *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 $M, \sigma \models I$.*

This lemma states that the definition of standard L-models is well-formed (i.e. the standard L-model of an L-normal model generator I is really an L-model of I). This lemma will be used (only) to prove the following expected theorem.

Expected Theorem 3.2.3 *The standard L-model of an L-normal model generator I is a least L-model of I.*

(We have a difficulty of calling the above assertions. Other ways are to call them *axioms* or *assumptions* or a *lemma/theorem to be proved*. The names "axiom" and "assumption" are not very suitable here, because one would not say "proof of an axiom" or "proof of an assumption".)

3.3 Fixpoint Semantics

We now return to the direct consequence operator $T_{L,P}$. 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 3.3.1 For L being one of the considered serial modal logics, define $\preceq_L$ to be the least reflexive and transitive relation between modal operators such that:

- $\Diamond_i \preceq_L \langle E\rangle_i \preceq_L \Box_i$ and $\Diamond_i \preceq_L \langle X\rangle_i \preceq_L \Box_i$,
- $\Box_i \preceq_L \Box_j$ and $\Diamond_j \preceq_L \Diamond_i$ if $L \in \{KDI4_s, KDI4, KDI4_s5, KDI45\}$ and $i \leq j$,

- $\Box_i \preceq_L \Box_j$ and $\Diamond_j \preceq_L \Diamond_i$ if $L = KD4I_g5_a$ and $g(i) \subseteq g(j)$,
- $\Box_i \preceq_L \Box_j$ and $\Diamond_j \preceq_L \Diamond_i$ if $L = sCFG$ and $\Box_j\varphi \rightarrow \Box_i\varphi$ is L-valid (for every φ).

By Corollary 2.5.2, the problem of checking whether $\nabla \preceq_L \nabla'$ for $L = sCFG$ is decidable.

Definition 3.3.2 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 $\Box_2\langle F\rangle_1$.

Expected Lemma 3.3.1 *If $\Box_{i_1} \ldots \Box_{i_h}$ is a $\Box$-lifting form of a modality $\triangle$ in L-normal labeled form and $\triangle$ is an L-instance of $\boxdot$, then $\boxdot\varphi \vDash_L \Box_{i_1} \ldots \Box_{i_h}\varphi$ for any formula φ without labeled modal operators.*

This lemma holds for the considered serial modal logics with $\preceq_L$ defined in Definition 3.3.1, because $\Box_{i_1} \ldots \Box_{i_h}$ is an L-instance of $\boxdot$.

Definition 3.3.3 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 $\boxdot'\varphi \rightarrow \boxdot\varphi$ 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.

Definition 3.3.4 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 3.3.5 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.

As an example, for $L = KDI4_s5$, the operator Sat_L is specified by three rules: (a) $\Box_i E \rightarrow \Box_j E$ if $i > j$, (b) $\Box_i E \rightarrow \Box_m\Box_i E$, (c) $\langle F\rangle_i E \rightarrow \Box_m\Diamond_i E$; and we have $Sat_L(\{\Box_2 p(a)\}) = \{\Box_2 p(a), \Box_1 p(a), \Box_m\Box_2 p(a), \Box_m\Box_1 p(a)\}$. (Recall that m is the maximal modal index.)

We expect the following property of Sat_L:

Expected Lemma 3.3.2 *Let I be an L-normal model generator, M the standard L-model of I, and α a ground L-MProlog goal atom. Suppose that $M \vDash \alpha$. Then α is an L-instance of some atom of $Sat_L(I)$.*

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 3.3.6 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 3.3.7 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 $\boxdot(A \leftarrow B_1, \ldots, B_n)$ is a ground L-instance of some program clause of P and $\triangle$ is a maximally general[5] 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 $1 \leq i \leq n$), 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 3.3.8 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 $NF_L(I) = I$.

As an example, for $L = KDI4_s5$, the operator NF_L is specified by the only rule: $\nabla\nabla' E \rightarrow \nabla' E$ if ∇' is of the form $\Box_i$ or $\langle E\rangle_i$; and we have $NF_L(\{\langle q(a)\rangle_1\langle p(a)\rangle_1 p(a)\}) = \{\langle p(a)\rangle_1 p(a)\}$.

Definition 3.3.9 Define $T_{L,P}(I) = NF_L(T_{^0L,P}(Sat_L(I)))$.

Lemma 3.3.3 *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. The operator $T_{L,P}$ is monotonic and compact because Sat_L, $T_{^0L,P}$ and NF_L are all increasingly monotonic and compact. It follows that $T_{L,P}$ is continuous. The second assertion of the lemma follows from the Kleen theorem. •

Notation 3.3.10 Denote the least fixpoint $T_{L,P} \uparrow \omega$ by $I_{L,P}$ and the standard L-model of $I_{L,P}$ by $M_{L,P}$.

Definition 3.3.11 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 3.3.1 Consider the following program P in $L = KDI4_s5$:

$$\Diamond_1 s(a) \leftarrow \qquad \Box_1(q(x) \leftarrow r(x), s(x))$$
$$\Box_1(\Box_1 r(x) \leftarrow s(x)) \qquad \Box_2(p(x) \leftarrow \Diamond_2 q(x))$$

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

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

We expect the following lemmas:

Expected Lemma 3.3.4 *If P is an L-MProlog program then $P \vDash_L I_{L,P}$.*

Expected Lemma 3.3.5 *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.*

Using the two above lemmas and Expected Theorem 3.2.3, we can derive:

Theorem 3.3.6 *For an L-MProlog program P, $M_{L,P}$ is a least L-model of P.*

Proof. By Lemma 3.3.5, $M_{L,P}$ is an L-model of P. Let M be an arbitrary L-model of P. By Lemma 3.3.4, $M \vDash I_{L,P}$. Hence, by Theorem 3.2.3, $M_{L,P} \leq M$. Therefore $M_{L,P}$ 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 3.4.1 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 3.4.2 Let $G = \leftarrow \alpha_1, \ldots, \alpha_i, \ldots, \alpha_k$ be a goal and $\varphi = \boxdot(A \leftarrow B_1, \ldots, B_n)$ a program clause. Then G' is *derived* from G and φ in L using an mgu θ, and called an *L-resolvent* of G and φ, if the following conditions hold:

- $\alpha_i = \triangle' A'$, with $\triangle'$ in L-normal labeled form, is called the *selected atom*, and A' is called the *selected head atom*;
- $\triangle'$ 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 3.4.3 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[6] atom variable. We call $\triangle\langle X\rangle_i E$ a *backward labeled form* of $\triangle\Diamond_i E$.

Definition 3.4.4 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 3.4.5 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".

As an example, for $L = KDI4_s5$, the operator rNF_L is specified by the only rule: $\nabla E \leftarrow \langle X\rangle_j \nabla E$ if ∇ is of the form $\Box_i$ or $\langle E\rangle_i$, and X is a fresh atom variable; and 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 3.4.6 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.

Definition 3.4.7 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 [44]). Each φ_i is called an *input clause/rule* of the derivation.

Definition 3.4.8 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.

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

Definition 3.4.9 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 $\theta_1 \dots \theta_n$ to the variables and atom variables of G, where $\theta_1, \dots, \theta_n$ is the sequence of mgu's used in an SLD-refutation of $P \cup \{G\}$ in L.

Example 3.4.1 Consider the following program P and the goal $G = \leftarrow \Box_1 p(x)$ in $L = KDI4_s5$:

$$\varphi_1 = \Box_2(p(x) \leftarrow \Diamond_2 q(x))$$
$$\varphi_2 = \Box_1(q(x) \leftarrow r(x), s(x))$$
$$\varphi_3 = \Box_1(\Box_1 r(x) \leftarrow s(x))$$
$$\varphi_4 = \Diamond_1 s(a) \leftarrow$$

Assume that the operators rNF_L and $rSat_L$ are specified by the following rules:

rNF_L : (a) $\nabla E \leftarrow \langle X \rangle_j \nabla E$ if ∇ is of the form $\Box_i$ or $\langle E \rangle_i$, and X is a fresh atom variable
$rSat_L$: (b) $\triangle \Diamond_i E \leftarrow \triangle \langle X \rangle_i E$ for X being a fresh atom variable
(c) $\triangle \nabla_i \alpha \leftarrow \triangle \Box_j \alpha$ if $i \leq j$
(d) $\triangle \Diamond_i E \leftarrow \triangle \Diamond_j E$ if $i > j$
(e) $\nabla \nabla' E \leftarrow \nabla' E$ if ∇' is of the form $\Box_i$ or $\Diamond_i$

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	$\{x_1/x\}$
$\leftarrow \Diamond_2 q(x)$	(e)	
$\leftarrow \Diamond_1 q(x)$	(d)	
$\leftarrow \langle X \rangle_1 q(x)$	(b)	
$\leftarrow \langle X \rangle_1 r(x), \langle X \rangle_1 s(x)$	φ_2	$\{x_5/x\}$
$\leftarrow \Box_1 r(x), \langle X \rangle_1 s(x)$	(c)	
$\leftarrow \langle Y \rangle_1 \Box_1 r(x), \langle X \rangle_1 s(x)$	(a)	
$\leftarrow \langle Y \rangle_1 s(x), \langle X \rangle_1 s(x)$	φ_3	$\{x_8/x\}$
$\leftarrow \langle X \rangle_1 s(a)$	φ_4	$\{x/a, Y/s(a)\}$
$\diamond$	φ_4	$\{X/s(a)\}$

3.5 Soundness and Completeness of SLD-Resolution

We prove soundness and completeness of SLD-resolution for L-MProlog using certain "expected" lemmas, which are strongly dependent on concrete instantiations of the framework for L. Informally, an SLD-resolution calculus is sound if every computed answer for $P \cup \{G\}$ is a correct answer for $P \cup \{G\}$, and is complete if for every correct answer for $P \cup \{G\}$ there exists a computed answer for $P \cup \{G\}$ that is more general.

Definition 3.5.1 We say that an atom β is *derivable* from α using $rSat_L$ (resp. (i) rNF_L, (ii) $rSat_L$ and rNF_L) if there exists a sequence of atoms $\alpha_0, \dots, \alpha_k$ with $k \geq 0$, $\alpha_0 = \alpha$ and $\alpha_k = \beta$ such that for every $1 \leq i \leq k$, $\alpha_i = rSat_L(\alpha_{i-1})$ (resp. (i) $\alpha_i =_{\theta_i} rNF_L(\alpha_{i-1})$ for some θ_i, (ii) $\alpha_i = rSat_L(\alpha_{i-1})$ or $\alpha_i =_{\theta_i} rNF_L(\alpha_{i-1})$ for some θ_i).

The main results are proved using the following expected properties of $rSat_L$ and rNF_L:

Expected Lemma 3.5.1 *Let $\triangle$ and $\triangle'$ be ground modalities in L-normal labeled form. Let B be an atom of the form E, $\Diamond_i E$, or $\Box_i E$, and B' an atom of the form E, $\Diamond_j E$, $\langle X \rangle_j E$, or $\Box_j E$, where X is a fresh atom variable. Suppose that $\triangle$ is an L-instance of $\triangle'$ and B is an L-instance of B'. Then $\triangle' B'$ is derivable from $\triangle B$ using $rSat_L$.*

Expected Lemma 3.5.2 *Suppose that β is an atom in almost L-normal labeled form and $\alpha \in Sat_L(\{\beta\})$ or $\alpha \in NF_L(\{\beta\})$. Then there exists an atom β' and a substitution θ s.t. $\beta = \beta'\theta$, the domain of θ consists of fresh atom variables, and β' is derivable from α using $rSat_L$ and rNF_L.*

Expected Lemma 3.5.3 *Let $\beta = rSat_L(\alpha)$, M be an L-model, σ a $\Diamond$-realization function on M, and θ a substitution. Suppose that $M, \sigma \vDash \forall_c(\beta'\theta)$ for some $\Box$-lifting form β' of β. Then $M, \sigma \vDash \forall_c(\alpha'\theta)$ for some $\Box$-lifting form α' of α.*

Expected Lemma 3.5.4 *Let $\beta =_\delta rNF_L(\alpha)$, M be an L-model, σ a maximal $\Diamond$-realization function on M, and θ a substitution. Suppose that $M, \sigma \vDash \forall_c(\beta'\theta)$ for some $\Box$-lifting form β' of β. Then $M, \sigma \vDash \forall_c(\alpha'\delta\theta)$ for some $\Box$-lifting form α' of α.*

3.5.1 Soundness

We first prove the following auxiliary lemma.

Lemma 3.5.5 *Let M be a Kripke model, σ a $\Diamond$-realization function on M, and θ a substitution. Suppose that $\triangle^{(1)}, \ldots, \triangle^{(l)}$ are $\Box$-lifting forms of $\triangle$ and $M, \sigma \vDash \forall_c((\triangle^{(1)} B_1 \wedge \ldots \wedge \triangle^{(l)} B_l)\theta)$. Then there exists the most general L-instance $\triangle'$ of $\triangle^{(1)}, \ldots, \triangle^{(l)}$, which is a $\Box$-lifting form of $\triangle$ and satisfies $M, \sigma \vDash \forall_c((\triangle' B_1 \wedge \ldots \wedge \triangle' B_l)\theta)$.*

Proof. Let $h = |\triangle|$ (the number of modal operators in $\triangle$). For $1 \leq j \leq l$ and $1 \leq k \leq h$, let $\nabla^{(j,k)}$ be the modal operator at position k of $\triangle^{(j)}$, and $\nabla^{(k)}$ the modal operator at position k of $\triangle$. Let i_k be the modal index (i.e. subscript) of the modal operator $\nabla^{(k)}$. If $\nabla^{(j,k)} = \Box_{i_k}$ for all $1 \leq j \leq l$, then let $\nabla^{(k')} = \Box_{i_k}$, else let $\nabla^{(k')} = \nabla^{(k)}$. Let $\triangle' = \nabla^{(1')} \ldots \nabla^{(h')}$. Clearly, $\triangle'$ is the most general L-instance of $\triangle^{(1)}, \ldots, \triangle^{(l)}$ and is a $\Box$-lifting form of $\triangle$.

Because that for $1 \leq j \leq l$, $\triangle^{(j)}$ is a $\Box$-lifting form of $\triangle'$ and $M, \sigma \vDash \forall_c((\triangle^{(j)} B_j)\theta)$, it can be proved by induction on k that $M, \sigma \vDash \forall_c((\nabla^{(1')} \ldots \nabla^{(k')} \top)\theta)$, for $1 \leq k \leq h$. It follows that $M, \sigma \vDash \forall_c((\triangle' \top)\theta)$. Because $\triangle^{(j)}$ is a $\Box$-lifting form of $\triangle'$, for $1 \leq j \leq l$, and $M, \sigma \vDash \forall_c((\triangle^{(1)} B_1 \wedge \ldots \wedge \triangle^{(l)} B_l)\theta)$, we conclude that $M, \sigma \vDash \forall_c((\triangle' B_1 \wedge \ldots \wedge \triangle' B_l)\theta)$. •

The soundness theorem is based on the following lemma:

Lemma 3.5.6 *Let P be an L-MProlog program and $G = \leftarrow \alpha_1, \ldots, \alpha_k$ be a goal. Then for every computed answer θ in L for $P \cup \{G\}$ there exists a goal $G' = \leftarrow \alpha'_1, \ldots, \alpha'_k$ such that α'_i is a $\Box$-lifting form of α_i, for $1 \leq i \leq k$, and $P \vDash_L \forall_c((\alpha'_1 \wedge \ldots \wedge \alpha'_k)\theta)$.*

Proof. Let M be an arbitrary L-model of P and σ a maximal $\Diamond$-realization function on M. Let the refutation of $P \cup \{G\}$ in L consist of a sequence $G_0 = G, G_1, \ldots, G_n$ of goals, a sequence $\varphi_1, \ldots, \varphi_n$ of variants of program clauses of P, $rSat_L$ rules, or rNF_L rules, and a sequence $\theta_1, \ldots, \theta_n$ of mgu's. Let θ be the computed answer. We prove by induction on n that for every $1 \leq i \leq k$ there exists a $\Box$-lifting form α'_i of α_i such that $M, \sigma \vDash \forall_c((\alpha'_1 \wedge \ldots \wedge \alpha'_k)\theta)$.

Suppose that $n = 1$. This means that $G = \leftarrow \alpha_1$ with $\alpha_1 = \triangle' A'$, A' is the selected head atom, and the empty clause is an L-resolvent of G and some input clause $\varphi_1 = \boxdot(A \leftarrow)$.

By Lemma 3.3.1, $P \models_L \forall(\Box_{i_1} \ldots \Box_{i_h} A)$, where $\Box_{i_1} \ldots \Box_{i_h}$ is a $\Box$-lifting form of $\triangle'$. If A' is of the form $\Box_i E$ or E, then $A'\theta_1 = A\theta_1$, and $P \models_L \forall(\Box_{i_1} \ldots \Box_{i_h} A'\theta_1)$. Suppose that $A' = \langle F \rangle_i E'$ or $A' = \langle X \rangle_i E'$. Thus $A = \Diamond_i E$. Let $A'' = \langle E \rangle_i E$ (the forward labeled form of A). We have $A'\theta_1 = A''\theta_1 = \langle E'' \rangle_i E''$ for some E''. Since $P \models_L \forall(\Box_{i_1} \ldots \Box_{i_h} A)$, we have $P \models_L \forall(\Box_{i_1} \ldots \Box_{i_h} \Diamond_i E'')$. It follows that $M, \sigma \models \forall_c(\Box_{i_1} \ldots \Box_{i_h} \langle E'' \rangle_i E'')$, because M is an L-model of P and σ is a maximal $\Diamond$-realization function on M. Hence $M, \sigma \models \forall_c(\Box_{i_1} \ldots \Box_{i_h} A'\theta_1)$. Thus, for $\alpha'_1 = \Box_{i_1} \ldots \Box_{i_h} A'$, we have $M, \sigma \models \forall_c(\alpha'_1\theta)$.

Next suppose that the result holds for computed answers which come from refutations of length less than n. There are the following cases: G_1 is derived from G and an $rSat_L/rNF_L$ rule variant, or G_1 is an L-resolvent of G and a variant of some program clause of P. The case G_1 is derived from G and an $rSat_L$ rule variant immediately follows from the inductive assumption and Lemma 3.5.3.

Suppose that G_1 is derived from G and an rNF_L rule variant, α_i is the selected atom and it is replaced by $\beta =_{\theta_1} rNF_L(\alpha_i)$. We have

$$G_1 = \leftarrow \alpha_1\theta_1, \ldots, \alpha_{i-1}\theta_1, \beta, \alpha_{i+1}\theta_1 \ldots, \alpha_k\theta_1$$

By the inductive assumption, there exist a $\Box$-lifting form α'_j of α_j, for $1 \leq j \leq k$ and $j \neq i$, and a $\Box$-lifting form β' of β such that

$$M, \sigma \models \forall_c((\alpha'_1\theta_1 \wedge \ldots \wedge \alpha'_{i-1}\theta_1 \wedge \beta' \wedge \alpha'_{i+1}\theta_1 \wedge \ldots \wedge \alpha'_k\theta_1)\theta_2 \ldots \theta_n)$$

We have $M, \sigma \models \forall_c(\beta'\theta_2 \ldots \theta_n)$. Hence, by Lemma 3.5.4, there exists a $\Box$-lifting form α'_i of α_i such that $M, \sigma \models \forall_c(\alpha'_i\theta_1\theta_2 \ldots \theta_n)$. Therefore $M, \sigma \models \forall_c((\alpha'_1 \wedge \ldots \wedge \alpha'_k)\theta)$.

Now suppose that G_1 is derived in L from G and an input clause $\varphi = \boxdot(A \leftarrow B_1, \ldots, B_l)$ $(l \geq 0)$, the selected atom is $\alpha_i = \triangle' A'$, and A' is the selected head atom. We have

$$G_1 = \leftarrow (\alpha_1, \ldots, \alpha_{i-1}, \triangle' B_1, \ldots, \triangle' B_l, \alpha_{i+1}, \ldots, \alpha_k)\theta_1$$

By the inductive assumption, there exists a goal

$$G'_1 = \leftarrow (\alpha'_1, \ldots, \alpha'_{i-1}, \triangle^{(1')} B'_1, \ldots, \triangle^{(l')} B'_l, \alpha'_{i+1}, \ldots, \alpha'_k)\theta_1$$

such that

$$M, \sigma \models \forall_c((\alpha'_1 \wedge \ldots \wedge \alpha'_{i-1} \wedge \triangle^{(1')} B'_1 \wedge \ldots \wedge \triangle^{(l')} B'_l \wedge \alpha'_{i+1} \wedge \ldots \wedge \alpha'_k)\theta)$$

where α'_j is a $\Box$-lifting form of α_j, for $1 \leq j \leq k$ and $j \neq i$, and $\triangle^{(j')} B'_j$ is a $\Box$-lifting form of $\triangle' B_j$ with $|\triangle^{(j')}| = |\triangle'|$, for $1 \leq j \leq l$. Let $\Box_{i_1} \ldots \Box_{i_h}$ be a $\Box$-lifting form of $\triangle'$, and $\triangle''$ be the most general L-instance of $\triangle^{(1')}$, $\ldots$, $\triangle^{(l')}$ if $l > 0$, which exists due to Lemma 3.5.5, and be $\Box_{i_1} \ldots \Box_{i_h}$ otherwise. By Lemma 3.5.5, $\triangle''$ is a $\Box$-lifting form of $\triangle'$, and $M, \sigma \models \forall_c((\triangle'' B'_1 \wedge \ldots \wedge \triangle'' B'_l)\theta)$ if $l > 0$. Since M is an L-model of P, by Lemma 3.3.1, we have $M \models \forall(\Box_{i_1} \ldots \Box_{i_h}(B_1 \wedge \ldots \wedge B_l \rightarrow A))$. Hence $M, \sigma \models \forall_c((\triangle'' A)\theta)$ (because $\Box_{i_1} \ldots \Box_{i_h}$ is a $\Box$-lifting form of $\triangle''$, B'_j is a $\Box$-lifting form of B_j, and L is a serial modal logic). Let A'' be the forward labeled form of A. Since σ is a maximal $\Diamond$-realization function on M, it follows that $M, \sigma \models \forall_c((\triangle'' A'')\theta)$. Since $A'\theta_1 = A''\theta_1$, by choosing $\alpha'_i = \triangle'' A'$, we have that α'_i is a $\Box$-lifting form of α_i and $M, \sigma \models \forall_c((\alpha'_1 \wedge \ldots \wedge \alpha'_k)\theta)$. This completes the proof. •

Theorem 3.5.7 (Soundness 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 $P \cup \{G\}$.*

Proof. Let $G = \leftarrow \alpha_1, \ldots, \alpha_k$, where each α_i is of the form $\boxdot E$ or $\boxdot\Diamond E$. Let θ be a computed answer in L for $P \cup \{G\}$. Since L is a serial modal logic, by Lemma 3.5.6, we have $P \models_L \forall_c((\alpha_1 \wedge \ldots \wedge \alpha_k)\theta)$. Assume that the signature contains enough constant symbols, for example, infinitely many. Then it follows that $P \models_L \forall((\alpha_1 \wedge \ldots \wedge \alpha_k)\theta)$. Hence θ is a correct answer in L for $P \cup \{G\}$. •

3.5.2 Completeness

We use a standard method to prove completeness of our SLD-resolution calculus (cf. [44, 42]). In general, completeness of a resolution calculus is first proved for the ground version and then lifted to the case with variables. The flow of this subsection follows Lloyd [44]. The proofs of Lemmas 3.5.8, 3.5.9, 3.5.14 and Theorem 3.5.15 are very similar to the ones given for classical logic programming in Lloyd's book, but we present all of them to make the work self-contained.

We first define unrestricted SLD-refutation and give the *mgu lemma* and the *lifting lemma*.

Definition 3.5.2 An *unrestricted SLD-refutation* in L is an SLD-refutation in L, except that we drop the requirement that the substitutions θ_i be most general unifiers. They are only required to be unifiers. In an unrestricted SLD-resolution, 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.

Lemma 3.5.8 (Mgu Lemma) *Let P be an L-MProlog program and G be a goal. Suppose that $P \cup \{G\}$ has an unrestricted SLD-refutation in L. Then $P \cup \{G\}$ has an SLD-refutation in L of the same length such that, if $\theta_1, \ldots, \theta_n$ are the unifiers from the unrestricted refutation and $\theta'_1, \ldots, \theta'_n$ are mgu's from the refutation, then there exists a substitution γ such that $\theta_1 \ldots \theta_n = \theta'_1 \ldots \theta'_n\gamma$.*

Proof. Let the unrestricted refutation of $P \cup \{G\}$ consist of a sequence $G_0 = G, G_1, \ldots, G_n$ of goals, a sequence $\varphi_1, \ldots, \varphi_n$ of variants of program clauses of P, $rSat_L$ rules, or rNF_L rules, and a sequence $\theta_1, \ldots, \theta_n$ of unifiers. We prove the result by induction on n.

Suppose that $n = 1$. This means that $G = \leftarrow \triangle' A'$ and the empty clause is an L-resolvent of G and the input clause $\varphi_1 = \boxdot(A \leftarrow)$, where A' is the selected head atom. Let θ'_1 be an mgu of A' and the forward labeled form of A. Then $\theta_1 = \theta'_1\gamma$ for some γ. Furthermore, $P \cup \{G\}$ has a refutation in L consisting of $G_0 = G$, $G_1 = \diamond$ (the empty goal) with input clause φ_1 and mgu θ'_1.

Now suppose that the result holds for unrestricted refutations with length less than n. Let $G = \leftarrow \alpha_1, \ldots, \alpha_k$ and α_i be the selected atom of G.

Suppose that G_1 is derived from G and the input clause $\varphi_1 = \boxdot(A \leftarrow B_1, \ldots, B_l)$ in L, the selected atom α_i is $\triangle' A'$, where A' is the selected head atom. There exists an mgu θ'_1 for A' and the forward labeled form of A. We have $\theta_1 = \theta'_1\delta$ for some δ. Let G'_1 be the goal derived in the same way as G_1 but with θ'_1 instead of θ_1. We have $G_1 = G'_1\delta$. Then G_2 can be derived from G'_1 in the same way as from G_1 but with unifier $\delta\theta_2$ instead of θ_2. Thus $P \cup \{G\}$ has an unrestricted refutation in L consisting of $G_0 = G, G'_1, G_2, \ldots, G_n$ with unifiers $\theta'_1, \delta\theta_2, \theta_3, \ldots, \theta_n$. By the inductive assumption, $P \cup \{G'_1\}$ has a refutation in L with mgu's $\theta'_2, \ldots, \theta'_n$ such that $\delta\theta_2 \ldots \theta_n = \theta'_2 \ldots \theta'_n\gamma$, for some γ. Thus $P \cup \{G\}$ has a refutation in L consisting of $G_0 = G, G'_1, \ldots, G'_n = \diamond$ with mgu's $\theta'_1, \theta'_2 \ldots \theta'_n$ such that $\theta_1\theta_2 \ldots \theta_n = \theta'_1\delta\theta_2 \ldots \theta_n = \theta'_1\theta'_2 \ldots \theta'_n\gamma$.

The cases when G_1 is derived from G and an $rSat_L/rNF_L$ rule variant are similar to the above case. •

Lemma 3.5.9 (Lifting Lemma) *Let P be an L-MProlog program, G a goal, and θ a substitution. Suppose there exists an SLD-refutation of $P \cup \{G\theta\}$ in L such that the variables of the input clauses are distinct from the variables in G and θ. Then there exists an SLD-refutation of $P \cup \{G\}$ in L of the same length such that, if $\theta_1, \ldots, \theta_n$ are the mgu's from the refutation of $P \cup \{G\theta\}$ and $\theta'_1, \ldots, \theta'_n$ are the mgu's from the refutation of $P \cup \{G\}$, then there exists a substitution γ such that $\theta\theta_1 \ldots \theta_n = \theta'_1 \ldots \theta'_n\gamma$.*

Proof. Let the refutation of $P \cup \{G\theta\}$ consist of a sequence $G_0 = G, G_1, \ldots, G_n$ of goals, a sequence $\varphi_1, \ldots, \varphi_n$ of variants of program clauses of P, $rSat_L$ rules, or rNF_L rules, and a sequence $\theta_1, \ldots, \theta_n$ of mgu's.

Suppose that G_1 is an L-resolvent of $G\theta$ and the input clause φ_1 using θ_1. Let $\varphi_1 = \boxdot(A \leftarrow B_1, \ldots, B_l)$, $G = \leftarrow \alpha_1, \ldots, \alpha_k$, and the selected atom of $G\theta$ be $\alpha_i\theta = (\triangle' A')\theta$, where $A'\theta$ is the selected head atom. We have that $\theta\theta_1$ is a unifier for A' and the forward labeled form of A. The result of resolving G and φ_1 using $\theta\theta_1$ is exactly G_1. Thus we obtain an unrestricted refutation of $P \cup \{G\}$ in L, which looks exactly like the given refutation of $P \cup \{G\theta\}$, except the original goal is different and the first unifier is $\theta\theta_1$. Now apply the mgu lemma.

The cases when G_1 is derived from G and an $rSat_L/rNF_L$ rule are similar to the above case. •

Lemma 3.5.10 (Weak Lifting Lemma) *Let P be an L-MProlog program, G a goal, and θ a substitution. Suppose there exists an SLD-refutation of $P \cup \{G\theta\}$ in L. Then there exists an SLD-refutation of $P \cup \{G\}$ in L.*

Proof. Change each variable x occurring in the input clauses of the refutation of $P \cup \{G\theta\}$ by x' that does not occur in G and θ. Then recompute the refutation. The resulting derivation will still be an SLD-refutation for $P \cup \{G\theta\}$ (possibly with different used mgu's and a different computed answer). Then apply the lifting lemma 3.5.9 for the new refutation. •

The following lemma is an essential part of the completeness proof.

Lemma 3.5.11 *Let P be an L-MProlog program and $\alpha \in I_{L,P}$. Then $P \cup \{\leftarrow \alpha\}$ has an SLD-refutation in L.*

Proof. We prove by induction on n that if $\alpha \in T_{L,P}\uparrow n$ then $P \cup \{\leftarrow \alpha\}$ has an SLD-refutation in L. This assertion obviously holds for $n = 0$, since $T_{L,P}\uparrow 0 = \emptyset$.

Suppose that the assertion holds for $(n-1)$ in the place of n. Let $\alpha \in T_{L,P}\uparrow n$. There exist a program clause $\varphi = \boxdot(A \leftarrow B_1, \ldots, B_k)$ of P, with $k \geq 0$, a substitution θ, modalities $\triangle'$ and $\boxdot'$, ground atoms $\gamma_1, \ldots, \gamma_k \in T_{L,P}\uparrow(n-1)$, and ground atoms $\beta_1, \ldots, \beta_k$, α' such that:

- $\beta_i \in Sat_L(\{\gamma_i\})$, for $1 \leq i \leq k$;
- $\beta_i = \triangle^{(i)} B'_i$ and $B_i\theta$ is an L-instance of B'_i, for $1 \leq i \leq k$;
- $\boxdot'$ is an L-context instance of $\boxdot$;
- $\triangle'$ is in the L-normal labeled form and is an L-instance of $\triangle^{(1)}, \ldots, \triangle^{(k)}, \boxdot'$;
- $\alpha' = \triangle' A'\theta$, where A' is the forward labeled form of A;

- $\alpha \in NF_L(\{\alpha'\})$.

By Lemma 3.5.2, there exist atoms α'', $\gamma'_1, \gamma'_2, \ldots, \gamma'_k$, and ground substitutions $\delta_0, \ldots, \delta_k$ with disjoint domains such that:

- α'' is derivable from α using $rSat_L$ and rNF_L, and $\alpha' = \alpha''\delta_0$,
- γ'_i is derivable from β_i using $rSat_L$ and rNF_L, and $\gamma_i = \gamma'_i\delta_i$, for $1 \leq i \leq k$.

Let $\delta = \delta_1 \ldots \delta_k$ if $k > 0$, and $\delta = \varepsilon$ otherwise. By the inductive assumption, $P \cup \{\leftarrow \gamma_i\}$ has a refutation in L, for $1 \leq i \leq k$. Since $\gamma'_i\delta = \gamma'_i\delta_i = \gamma_i$, it follows that $P \cup \{\leftarrow \gamma'_i\delta\}$ has a refutation in L. Hence $P \cup \{\leftarrow (\gamma'_1, \ldots, \gamma'_k)\delta\}$ has a refutation in L, since $\gamma'_i\delta$ are ground. By the weak lifting lemma, $P \cup \{\leftarrow \gamma'_1, \ldots, \gamma'_k\}$ has a refutation in L. Since γ'_i is derivable from β_i using $rSat_L$ and rNF_L, it follows that $P \cup \{\leftarrow \beta_1, \ldots, \beta_k\}$ has a refutation in L.

For $1 \leq i \leq k$, if B'_i is of the form $\langle F\rangle_h E$ then let $\beta'_i = \triangle^{(i)}\langle X\rangle_h E$ and $\theta_i = \{X/F\}$, where X is a fresh atom variable; else let $\beta'_i = \beta_i$ and $\theta_i = \varepsilon$. Let $\theta' = \theta_1 \ldots \theta_k$ if $k > 0$, and $\theta' = \varepsilon$ otherwise. Since $\beta_i = \beta'_i\theta'$, $P \cup \{\leftarrow (\beta'_1, \ldots, \beta'_k)\theta'\}$ has a refutation in L. Hence, by the weak lifting lemma, $P \cup \{\leftarrow \beta'_1, \ldots, \beta'_k\}$ has a refutation in L. Therefore, by Lemma 3.5.1, $P \cup \{\leftarrow \triangle'B_1\theta, \ldots, \triangle'B_k\theta\}$ has a refutation in L. The goal $\leftarrow \triangle'B_1\theta, \ldots, \triangle'B_k\theta$ is an unrestricted L-resolvent of $\leftarrow \alpha'$ and φ. Hence, by the mgu lemma, $P \cup \{\leftarrow \alpha'\}$ has a refutation in L. This means that $P \cup \{\leftarrow \alpha''\delta_0\}$ has a refutation in L. By the weak lifting lemma, $P \cup \{\leftarrow \alpha''\}$ has a refutation in L. Since α'' is derivable from α using $rSat_L$ and rNF_L, we conclude that $P \cup \{\leftarrow \alpha\}$ has a refutation in L. •

Corollary 3.5.12 *Let P be an L-MProlog program and $\alpha \in Sat_L(I_{L,P})$. Then $P \cup \{\leftarrow \alpha\}$ has an SLD-refutation in L.*

Proof. There exists $\beta \in I_{L,P}$ such that $\alpha \in Sat_L(\{\beta\})$. By Lemma 3.5.2, there exist an atom β' and a substitution θ such that $\beta = \beta'\theta$ and β' is derivable from α using $rSat_L$ and rNF_L. Since $\beta \in I_{L,P}$, by Lemma 3.5.11, $P \cup \{\leftarrow \beta\}$ has a refutation in L. This means that $P \cup \{\leftarrow \beta'\theta\}$ has a refutation in L. By the weak lifting lemma, $P \cup \{\leftarrow \beta'\}$ has a refutation in L. Consequently, $P \cup \{\leftarrow \alpha\}$ has a refutation in L. •

Lemma 3.5.13 *Let P be an L-MProlog program and α a ground L-MProlog goal atom such that $M_{L,P} \models \alpha$. Then $P \cup \{\leftarrow \alpha\}$ has an SLD-refutation in L.*

Proof. By Lemma 3.3.2, α is an L-instance of some $\alpha' \in Sat_L(I_{L,P})$. By Corollary 3.5.12, $P \cup \{\leftarrow \alpha'\}$ has an SLD-refutation in L. If α' is of the form E, $\triangle\Diamond_i E$, or $\triangle\Box_i E$ then, by Lemma 3.5.1, $P \cup \{\leftarrow \alpha\}$ has an SLD-refutation in L. If α' is of the form $\triangle\langle F\rangle_i E$ then, by the weak lifting lemma, $P \cup \{\leftarrow \triangle\langle X\rangle_i E\}$ has an SLD-refutation in L, where X is a fresh atom variable. By the assumption about $\preceq_L$, α is also an L-instance of $\triangle\langle X\rangle_i E$. Hence, by Lemma 3.5.1, $P \cup \{\leftarrow \alpha\}$ has an SLD-refutation in L. •

For the main theorem, we need also the following auxiliary lemma.

Lemma 3.5.14 *Let P be an L-MProlog program and α an L-MProlog goal atom. Suppose that $\forall(\alpha)$ is a logical consequence in L of P. Then there exists an SLD-refutation of $P \cup \{\leftarrow \alpha\}$ in L with the identity substitution as the computed answer.*

Proof. Suppose α has variables $x_1, \ldots, x_n$. Let $a_1, \ldots, a_n$ be distinct constant symbols not appearing in P and α, and let θ be the substitution $\{x_1/a_1, \ldots, x_n/a_n\}$. Then it is clear that $\alpha\theta$ is a logical consequence in L of P. By Lemma 3.3.5, we have $M_{L,P} \vDash \alpha\theta$. Since $\alpha\theta$ is ground, by Lemma 3.5.13, $P \cup \{\leftarrow \alpha\theta\}$ has a refutation in L. Since the a_i do not appear in P or α, by replacing a_i by x_i (for $1 \leq i \leq n$) in this refutation, we obtain a refutation of $P \cup \{\leftarrow \alpha\}$ in L with the identity substitution as the computed answer. •

Theorem 3.5.15 (Completeness of SLD-Resolution) *Let P be an L-MProlog program and G an L-MProlog goal. 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.*

Proof. Suppose G is the goal $\leftarrow \alpha_1, \ldots, \alpha_k$. Since θ is a correct answer in L for $P \cup \{G\}$, $\forall((\alpha_1 \wedge \ldots \wedge \alpha_k)\theta)$ is a logical consequence of P in L. By Lemma 3.5.14, there exists a refutation of $P \cup \{\leftarrow \alpha_i\theta\}$ in L such that the computed answer is the identity substitution, for $1 \leq i \leq k$. We can combine these refutations into a refutation of $P \cup \{G\theta\}$ such that the computed answer is the identity substitution. Change each variable x occurring in the input clauses of the refutation of $P \cup \{G\theta\}$ by x' that does not occur in G and θ. Then recompute the refutation in the way such that when unifying $\{x, x'\}$ the substitution $\{x'/x\}$ (x' is changed to x) is used instead of $\{x/x'\}$. The resulting derivation will still be an SLD-refutation for $P \cup \{G\theta\}$ with the identity substitution as the computed answer. Applying the lifting lemma, we conclude that there exists a refutation of $P \cup \{G\}$ in L with computed answer γ such that $G\theta = G\gamma\delta$, for some substitution δ. •

3.5.3 Strong Completeness

In this subsection, we introduce the notion of selection rule, which is used to select atoms in SLD-derivations. We prove that SLD-resolution for L-MProlog using any selection rule is complete.

Definition 3.5.3 A *selection rule* is a function that maps an SLD-derivation $G_0, G_1, \ldots, G_n$ with $G_n = \leftarrow \alpha_1, \ldots, \alpha_k$ to some atom α_i $(1 \leq i \leq k)$.

Definition 3.5.4 Let P be an L-MProlog program, G a goal, and $\mathcal{R}$ a selection rule. An *SLD-refutation* of $P \cup \{G\}$ in L *via* $\mathcal{R}$ is an *SLD-refutation* of $P \cup \{G\}$ in L that uses $\mathcal{R}$ to select atoms.

Theorem 3.5.16 (Strong Completeness of SLD-Resolution) *Let P be an L-MProlog program, G an L-MProlog goal, and $\mathcal{R}$ a selection rule. For every correct answer θ in L of $P \cup \{G\}$, there exists an SLD-refutation of $P \cup \{G\}$ in L via $\mathcal{R}$ with computed answer γ such that $G\theta = G\gamma\delta$ for some substitution δ.*

Sketch. Since θ is a correct answer in L for $P \cup \{G\}$, by the completeness theorem, there exists an unrestricted SLD-refutation of $P \cup \{G\theta\}$ in L in which every goal is ground. For such a refutation, any switching for a selection of an atom can be done without effects on the essence of the refutation. Let $\mathbf{R}$ be the set of all unrestricted SLD-refutations of $P \cup \{G\theta\}$ in L obtained from the mentioned unrestricted refutation by any combination of switching operations. Then let $\mathbf{R}'$ be the set of all SLD-refutations of $P \cup \{G\}$ in L obtained from the unrestricted refutations belonging to $\mathbf{R}$ by the lifting technique. The selection function $\mathcal{R}$ must be used by some refutation belonging to $\mathbf{R}'$. •

3.6 Summary

We have given a framework for developing fixpoint semantics, the least model semantics, and SLD-resolution calculi for L-MProlog programs. The base logic L is required to be a serial normal modal logic such that the L-frame restrictions consist of $\forall x \exists y\, R_i(x, y)$ (seriality), for $1 \leq i \leq m$, and some classical first-order Horn clauses.

Definition 3.6.1 By a *schema for semantics of L-MProlog* we mean a table consisting of a definition of L-normal form of modalities, a definition of $\preceq_L$, and rules specifying the operators Ext_L, Sat_L, NF_L, rNF_L, $rSat_L$. We say that such a schema is *correct* if all the expected results of this chapter hold for L-MProlog w.r.t. that schema.

To show correctness of a schema, we have to prove Expected Theorem 3.2.3 and Expected Lemmas 3.2.2, 3.3.1, 3.3.2, 3.3.4, 3.3.5, 3.5.1 – 3.5.4. Theorem 3.3.6 has been proved using Expected Theorem 3.2.3 and Expected Lemmas 3.3.4 and 3.3.5. It states that the fixpoint semantics coincides with the least model semantics. Theorems 3.5.7, 3.5.15, and 3.5.16 about soundness and completeness of SLD-resolution for L-MProlog has been proved using Expected Lemmas 3.3.1, 3.3.2, 3.3.5, 3.5.1 – 3.5.4.

Chapter 4

Instantiations of the Framework

4.1 A Schema for Semantics of $BSMM$-MProlog

In this section, let L be a $BSMM$ logic. In Table 4.1, we present a schema for semantics of $BSMM$-MProlog. The first rule specifying $rSat_L$ is a generalized version of the backward labeling rule and is dual to the first rule specifying Sat_L. The remaining rules specifying Sat_L and $rSat_L$ directly come from the axioms. This gives an impression that the schema relies on syntactic properties of the base logic. Clarity of the rules suggests a general method for translating axioms of a given modal logic into an SLD-resolution calculus for that logic.

Example 4.1.1 Consider the multimodal logic L specified by $m = 2$ (the number of different modal indices), $AD = \{1, 2\}$, $AT = \{1\}$, $AI = \{(2, 1)\}$, and $AB = A4 = A5 = \emptyset$. In other words, the logic is characterized by the axioms: $\Box_1\varphi \to \Diamond_1\varphi$; $\Box_2\varphi \to \Diamond_2\varphi$; $\Box_1\varphi \to \varphi$; and $\Box_2\varphi \to \Box_1\varphi$. Consider the following program P:

$$\begin{array}{l}\varphi_1 = \Diamond_2 p(a) \leftarrow \\ \varphi_2 = \Box_2(\Box_1 q(x) \leftarrow \Diamond_2 p(x)) \\ \varphi_3 = \Box_2(r(x) \leftarrow p(x), q(x))\end{array}$$

We have $T_{L,P}{\uparrow}1 = \{\langle p(a)\rangle_2 p(a)\}$ and

$$Sat_L(T_{L,P}{\uparrow}1) = \{\langle p(a)\rangle_2 p(a), \langle p(a)\rangle_2\Diamond_1 p(a), \langle p(a)\rangle_2\Diamond_2 p(a)\}$$

Applying the program clause φ_2 and its L-instance $\Box_1 q(x) \leftarrow \Diamond_2 p(x)$ to $Sat_L(T_{L,P}{\uparrow}1)$, we obtain $T_{L,P}{\uparrow}2 = T_{L,P}{\uparrow}1 \cup \{\langle p(a)\rangle_2\Box_1 q(a),\ \Box_1 q(a)\}$. The set $Sat_L(T_{L,P}{\uparrow}2)$ contains both $\langle p(a)\rangle_2 p(a)$ and $\langle p(a)\rangle_2 q(a)$. Hence, by applying φ_3, we have $\langle p(a)\rangle_2 r(a) \in T_{L,P}{\uparrow}3$ and arrive at

$$T_{L,P}{\uparrow}\omega = T_{L,P}{\uparrow}3 = \{\langle p(a)\rangle_2 p(a), \langle p(a)\rangle_2\Box_1 q(a), \Box_1 q(a), \langle p(a)\rangle_2 r(a)\}$$

We give below an SLD-refutation of $P \cup \{\leftarrow \Diamond_2 r(x)\}$ in L with computed answer $\{x/a\}$.

Goals	Input clauses/rules	MGUs
$\leftarrow \Diamond_2 r(x)$		
$\leftarrow \langle X\rangle_2 r(x)$	(1): $\triangle\Diamond_i\alpha \leftarrow \triangle\langle X\rangle_i\alpha$	
$\leftarrow \langle X\rangle_2 p(x), \langle X\rangle_2 q(x)$	$\Box_2(r(x) \leftarrow p(x), q(x))$	$\{x_2/x\}$
$\leftarrow \langle p(a)\rangle_2 q(a)$	$\Diamond_2 p(a) \leftarrow$	$\{X/p(a), x/a\}$
$\leftarrow \langle p(a)\rangle_2\Box_1 q(a)$	(3): $\triangle\alpha \leftarrow \triangle\Box_1\alpha$	
$\leftarrow \langle p(a)\rangle_2\Diamond_2 p(a)$	$\Box_2(\Box_1 q(x) \leftarrow \Diamond_2 p(x))$	$\{x_5/a\}$

$L = BSMM$, L-MProlog
$\preceq_L$ is defined by Definition 3.3.1 at page 34. No restrictions on L-normal form of modalities. No rules specifying NF_L and rNF_L. Rules specifying Ext_L and Sat_L: $\triangle\langle E\rangle_i\alpha \to \triangle\Diamond_i\alpha$ (1) $\triangle\Box_i\alpha \to \triangle\Diamond_i\alpha$ (2) $\triangle\Box_i\alpha \to \triangle\alpha$ if $AT(i)$ (3) $\triangle\alpha \to \triangle\Diamond_i\alpha$ if $AT(i)$ (4) $\triangle\Box_i\alpha \to \triangle\Box_j\alpha$ if $AI(i,j)$ (5) $\triangle\Diamond_j\alpha \to \triangle\Diamond_i\alpha$ if $AI(i,j)$ (6) $\triangle\alpha \to \triangle\Box_i\Diamond_j\alpha$ if $AB(i,j)$ (7) $\triangle\Diamond_i\Box_j\alpha \to \triangle\alpha$ if $AB(i,j)$ (8) $\triangle\Box_i\alpha \to \triangle\Box_j\Box_k\alpha$ if $A4(i,j,k)$ (9) $\triangle\Diamond_j\Diamond_k\alpha \to \triangle\Diamond_i\alpha$ if $A4(i,j,k)$ (10) $\triangle\Diamond_i\alpha \to \triangle\Box_j\Diamond_k\alpha$ if $A5(i,j,k)$ (11) $\triangle\Diamond_j\Box_k\alpha \to \triangle\Box_i\alpha$ if $A5(i,j,k)$ (12) Rules specifying $rSat_L$: $\triangle\Diamond_i\alpha \leftarrow \triangle\langle X\rangle_i\alpha$ where X is a fresh atom variable (1) $\triangle\nabla_i\alpha \leftarrow \triangle\Box_i\alpha$ (2) plus a rule $\alpha \leftarrow \beta$ for each k-th rule $\beta \to \alpha$ specifying Sat_L, $k \geq 3$, with the same accompanying condition (3)..(12)

Table 4.1: A schema for semantics of $BSMM$-MProlog

$\leftarrow \langle p(a)\rangle_2\Diamond_1 p(a)$	(6): $\triangle\Diamond_2\alpha \leftarrow \triangle\Diamond_1\alpha$
$\leftarrow \langle p(a)\rangle_2 p(a)$	(4): $\triangle\Diamond_1\alpha \leftarrow \triangle\alpha$
$\diamond$	$\Diamond_2 p(a) \leftarrow$

Theorem 4.1.1 *The schema given in Table 4.1 for semantics of BSMM-MProlog is correct.*

To prove this theorem we have to prove Expected Theorem 3.2.3 and Expected Lemmas 3.2.2, 3.3.2, 3.3.4, 3.3.5, 3.5.1 – 3.5.4. (Expected Lemma 3.3.1 clearly holds.) To do this we need *extended L-model graphs* (defined below) and some properties of them.

Definition 4.1.1 Let I be a model generator. Define Ext'_L to be the operator such that $Ext'_L(I)$ is the least set of atoms extending I and closed w.r.t. the rules specifying Sat_L. (Note that we allow $Ext'_L(I)$ to contain atoms not in almost labeled form and have that $Ext_L(I) \subseteq Sat_L(I) \subseteq Ext'_L(I)$.) The *extended L-model graph* of I is defined in the same way as the standard L-model graph of I but with $Ext'_L(I)$ in the place of $Ext_L(I)$.

Lemma 4.1.2 *Let I be a model generator, M the standard L-model graph of I, and M' the extended L-model graph of I. Then M' has the same frame as M, and furthermore, if $M =$*

$\langle W, \tau, R_1, \ldots, R_m, H\rangle$ and $M' = \langle W, \tau, R_1, \ldots, R_m, H'\rangle$ then for every $w \in W$, $H(w) \subseteq H'(w)$ and $H'(w) - H(w)$ is a set of formulas containing some unlabeled existential modal operators.

The proof of this lemma is straightforward.

The following lemma is similar to Lemma 3.2.1 and can also be proved by induction on the length of $\triangle$ in a straightforward way.

Lemma 4.1.3 *Let I be a model generator and $M = \langle W, \tau, R_1, \ldots, R_m, H\rangle$ be the extended L-model graph of I. Let $w = \langle E_1\rangle_{i_1} \ldots \langle E_k\rangle_{i_k}$ be a world of M and $\triangle = w$ be a modality. Then for α (resp. A)[1] not containing $\top$, $\alpha \in H(w)$ (resp. $A \in H(w)$) iff there exists a $\Box$-lifting form $\triangle'$ of $\triangle$ such that $\triangle'\alpha \in Ext'_L(I)$ (resp. $\triangle' A \in Sat_L(I)$).*

We give below the main lemma concerning extended L-model graphs.

Lemma 4.1.4 *Let I be a model generator and $M = \langle W, \tau, R_1, \ldots, R_m, H\rangle$ be the extended L-model graph of I. Then for any w and u such that $R_i(w, u)$ holds: i) if $\Box_i\alpha \in H(w)$ then $\alpha \in H(u)$, ii) if $\alpha \in H(u)$ then $\Diamond_i\alpha \in H(w)$.*

Proof. Let $(R'_j)_{1\le j\le m}$ be the skeleton of M. We prove this lemma by induction on the number of steps needed to obtain $R_i(w, u)$ when extending $(R'_j)_{1\le j\le m}$ to $(R_j)_{1\le j\le m}$.

Consider the first assertion. Suppose that $\Box_i\alpha \in H(w)$. By Lemma 4.1.3, there exists a $\Box$-lifting form $\triangle$ of w such that $\triangle\Box_i\alpha \in Ext'_L(I)$. Since $R_i(w, u)$ holds, there are the following cases to consider:

- Case $u = w\langle E\rangle_i$ and $R'_i(w, w\langle E\rangle_i)$: The assertion holds by the definition of M.
- Case $AT(i)$ holds and $u = w$: Since $\triangle\Box_i\alpha \in Ext'_L(I)$, we have $\triangle\alpha \in Ext'_L(I)$, and by Lemma 4.1.3, $\alpha \in H(u)$.
- Case $AI(i, j)$ holds and $R_i(w, u)$ is created from $R_j(w, u)$: Since $\triangle\Box_i\alpha \in Ext'_L(I)$, we have $\triangle\Box_j\alpha \in Ext'_L(I)$, and by Lemma 4.1.3, $\Box_j\alpha \in H(w)$. Hence, by the inductive assumption, $\alpha \in H(u)$.
- Case $AB(j, i)$ holds and $R_i(w, u)$ is created from $R_j(u, w)$: Since $\Box_i\alpha \in H(w)$, by the inductive assumption, $\Diamond_j\Box_i\alpha \in H(u)$. By Lemma 4.1.3, there exists a $\Box$-lifting form $\triangle'$ of u such that $\triangle'\Diamond_j\Box_i\alpha \in Ext'_L(I)$. Thus $\triangle'\alpha \in Ext'_L(I)$. Hence, by Lemma 4.1.3, $\alpha \in H(u)$.
- Case $A4(i, j, k)$ holds and $R_i(w, u)$ is created from $R_j(w, v)$ and $R_k(v, u)$: Since $\triangle\Box_i\alpha \in Ext'_L(I)$, we have $\triangle\Box_j\Box_k\alpha \in Ext'_L(I)$, and by Lemma 4.1.3, $\Box_j\Box_k\alpha \in H(w)$. Hence, by the inductive assumption, $\Box_k\alpha \in H(v)$ and $\alpha \in H(u)$.
- Case $A5(j, k, i)$ holds and $R_i(w, u)$ is created from $R_j(v, u)$ and $R_k(v, w)$: Since $\Box_i\alpha \in H(w)$, by the inductive assumption, $\Diamond_k\Box_i\alpha \in H(v)$. Hence, by Lemma 4.1.3, there exists a $\Box$-lifting form $\triangle'$ of v such that $\triangle'\Diamond_k\Box_i\alpha \in Ext'_L(I)$. Hence $\triangle'\Box_j\alpha \in Ext'_L(I)$, and by Lemma 4.1.3, $\Box_j\alpha \in H(v)$. By the inductive assumption, it follows that $\alpha \in H(u)$.

Consider the second assertion. Suppose that $\alpha \in H(u)$. By Lemma 4.1.3, there exists a $\Box$-lifting form $\triangle$ of u such that $\triangle\alpha \in Ext'_L(I)$. Since $R_i(w, u)$ holds, there are the following cases to consider:

[1]Recall that α denotes an atom of the form $\triangle'' E$, while A denotes a simple atom of the form E or ∇E, where E is a classical atom and ∇ is a modal operator.

- Case $u = w\langle E\rangle_i$ and $R'_i(w, w\langle E\rangle_i)$: Since $\alpha \in H(u)$, either $\Box_i\alpha$ or $\langle E\rangle_i\alpha$ belongs to $H(w)$. By Lemma 4.1.3, there exists a $\Box$-lifting form $\triangle'$ of w such that $\triangle'\Box_i\alpha \in Ext'_L(I)$ or $\triangle'\langle E\rangle_i\alpha \in Ext'_L(I)$. Hence $\triangle'\Diamond_i\alpha \in Ext'_L(I)$, and by Lemma 4.1.3, $\Diamond_i\alpha \in H(w)$.

- Case $AT(i)$ holds and $u = w$: Since $\triangle\alpha \in Ext'_L(I)$, $\triangle\Diamond_i\alpha \in Ext'_L(I)$, and by Lemma 4.1.3, $\Diamond_i\alpha \in H(w)$.

- Case $AI(i, j)$ holds and $R_i(w, u)$ is created from $R_j(w, u)$: Since $\alpha \in H(u)$, by the inductive assumption, $\Diamond_j\alpha \in H(w)$. By Lemma 4.1.3, there exists a $\Box$-lifting form $\triangle'$ of w such that $\triangle'\Diamond_j\alpha \in Ext'_L(I)$. Thus $\triangle'\Diamond_i\alpha \in Ext'_L(I)$. Hence, by Lemma 4.1.3, $\Diamond_i\alpha \in H(w)$.

- Case $AB(j, i)$ holds and $R_i(w, u)$ is created from $R_j(u, w)$: Since $\triangle\alpha \in Ext'_L(I)$, we have $\triangle\Box_j\Diamond_i\alpha \in Ext'_L(I)$. Hence, by Lemma 4.1.3, $\Box_j\Diamond_i\alpha \in H(u)$. By the inductive assumption, it follows that $\Diamond_i\alpha \in H(w)$.

- Case $A4(i, j, k)$ holds and $R_i(w, u)$ is created from $R_j(w, v)$ and $R_k(v, u)$: Since $\alpha \in H(u)$, by the inductive assumption, $\Diamond_k\alpha \in H(v)$ and $\Diamond_j\Diamond_k\alpha \in H(w)$. By Lemma 4.1.3, there exists a $\Box$-lifting form $\triangle'$ of w such that $\triangle'\Diamond_j\Diamond_k\alpha \in Ext'_L(I)$. Thus $\triangle'\Diamond_i\alpha \in Ext'_L(I)$, and by the Lemma 4.1.3, $\Diamond_i\alpha \in H(w)$.

- Case $A5(j, k, i)$ holds and $R_i(w, u)$ is created from $R_j(v, u)$ and $R_k(v, w)$: Since $\alpha \in H(u)$, by the inductive assumption, $\Diamond_j\alpha \in H(v)$. By Lemma 4.1.3, there exists a $\Box$-lifting form $\triangle'$ of v such that $\triangle'\Diamond_j\alpha \in Ext'_L(I)$. Thus $\triangle'\Box_k\Diamond_i\alpha \in Ext'_L(I)$, and by the Lemma 4.1.3, $\Box_k\Diamond_i\alpha \in H(v)$. Hence, by the inductive assumption, $\Diamond_i\alpha \in H(w)$.

•

To increase readability we recall expected lemmas and theorems before giving their proofs.

Expected Lemma 3.2.2 *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 $M, \sigma \vDash I$.*

Proof. By the definition, M is an L-model. Let $M' = \langle W, \tau, R_1, \ldots, R_m, H\rangle$ be the extended L-model graph of I. It can be proved by induction on the length of α that for any $w \in W$, if $\alpha \in H(w)$, then $M', \sigma, w \vDash \alpha$. The cases when α is a classical atom or $\alpha = \langle E\rangle_i\beta$ are trivial. The case when $\alpha = \Box_i\beta$ is solved by Lemmas 4.1.2 and 4.1.4. Hence $M, \sigma \vDash I$. •

Expected Theorem 3.2.3 *The standard L-model of an L-normal model generator I is a least L-model of I.*

Proof. Let $M = \langle W, \tau, R_1, \ldots, R_m, H\rangle$ be the standard L-model graph of I, σ the standard $\Diamond$-realization function, and $(R'_i)_{1\le i\le m}$ the skeleton of the standard L-model of I. By Lemma 3.2.2, M is an L-model of I. Let $N = \langle D, W_2, \tau_2, S_1, \ldots, S_m, \pi\rangle$ be an arbitrary L-model of I and σ_2 a $\Diamond$-realization function on N such that $N, \sigma_2 \vDash I \cup Serial_L$.

Let $r \subseteq W \times W_2$ be the least relation such that, for all w, w_2, u_2, E, i:

- $r(\tau, \tau_2)$;

- if $r(w, w_2)$ and $R'_i(w, w\langle E\rangle_i)$ hold, and $\sigma_2(w_2, \langle E\rangle_i)$ is defined, then $r(w\langle E\rangle_i, \sigma_2(w_2, \langle E\rangle_i))$;

- if $r(w, w_2)$ and $S_i(w_2, u_2)$ hold, then $r(w\langle\top\rangle_i, u_2)$.

Note that if $r(w, w_2)$ and $S_i(w_2, u_2)$ hold, then for $u = w\langle\top\rangle_i$ we have $r(u, u_2)$ and $R_i(w, u)$.

We prove that $M \leq_r N$. We first show that if $r(u, u_2)$ and $\alpha \in H(u)$ then $N, \sigma_2, u_2 \vDash \alpha$. We prove this by induction on the length of u. Suppose that $r(u, u_2)$ holds and $\alpha \in H(u)$. The case $u = \epsilon$ is trivial. Let $u = w\langle E\rangle_i$ and inductively assume that the assertion holds when u is replaced by w. There are two cases:

- $u_2 = \sigma_2(w_2, \langle E\rangle_i)$, $r(w, w_2)$, and $R'_i(w, w\langle E\rangle_i)$, for some $w_2 \in W_2$; or
- $E = \top$, $r(w, w_2)$, and $S_i(w_2, u_2)$, for some $w_2 \in W_2$.

Consider the first case. Since $\alpha \in H(u)$, either $\Box_i\alpha \in H(w)$ or $\langle E\rangle_i\alpha \in H(w)$. By the inductive assumption, either $N, \sigma_2, w_2 \vDash \Box_i\alpha$ or $N, \sigma_2, w_2 \vDash \langle E\rangle_i\alpha$. Hence, $N, \sigma_2, \sigma_2(w_2, \langle E\rangle_i) \vDash \alpha$, which means that $N, \sigma_2, u_2 \vDash \alpha$.

Consider the second case. Since $\alpha \in H(u)$, it follows that $\Box_i\alpha \in H(w)$. By the inductive assumption, $N, \sigma_2, w_2 \vDash \Box_i\alpha$, and hence $N, \sigma_2, u_2 \vDash \alpha$ since $S_i(w_2, u_2)$.

We now show that if $r(w, w_2)$ and $R'_i(w, w\langle E\rangle_i)$ hold then $\sigma_2(w_2, \langle E\rangle_i)$ is defined. The case $E = \top$ is trivial. Suppose that $r(w, w_2)$ and $R'_i(w, w\langle E\rangle_i)$ hold and $E \neq \top$. Thus, there exists $\langle E\rangle_i\alpha \in H(w)$ for some α. Hence $N, \sigma_2, w_2 \vDash \langle E\rangle_i\alpha$ and $\sigma_2(w_2, \langle E\rangle_i)$ is defined. Therefore, the second condition in the above definition of r can be simplified to "if $r(w, w_2)$ and $R'_i(w, w\langle E\rangle_i)$ hold then $r(w\langle E\rangle_i, \sigma_2(w_2, \langle E\rangle_i))$".

It is straightforward to prove by induction on the number of steps needed to obtain $R_i(w, u)$ when extending $(R'_j)_{1\leq j\leq m}$ to $(R_j)_{1\leq j\leq m}$ that if $R_i(w, u)$ then: i) if $r(w, w_2)$ then there exists u_2 such that $r(u, u_2)$ and $S_i(w_2, u_2)$; ii) if $r(u, u_2)$ then there exists w_2 such that $r(w, w_2)$ and $S_i(w_2, u_2)$. We give here only the base case, when $u = w\langle E\rangle_i$: i) Suppose that $r(w, w_2)$ holds. We have $R'_i(w, w\langle E\rangle_i)$, hence $\sigma_2(w_2, \langle E\rangle_i)$ is defined. The assertion holds for $u_2 = \sigma_2(w_2, \langle E\rangle_i)$. ii) Suppose that $r(u, u_2)$ holds. By the definition of r, there exists w_2 such that $r(w, w_2)$ and ($S_i(w_2, u_2)$ or $u_2 = \sigma_2(w_2, \langle E\rangle_i)$). It is clear that the assertion holds for such w_2.

We have proved that r satisfies all the conditions to guarantee $M \leq_r N$. This together with Lemma 3.2.2 implies that M is a least L-model of I. •

Expected Lemma 3.3.2 *Let I be an L-normal model generator, M the standard L-model of I, and α a ground L-MProlog goal atom. Suppose that $M \vDash \alpha$. Then α is an L-instance of some atom of $Sat_L(I)$.*

Proof. Let $M' = \langle W, \tau, R_1, \ldots, R_m, H\rangle$ be the extended L-model graph of I, $\boxdot = \Box_{i_1} \ldots \Box_{i_k}$ and $w = \langle\top\rangle_{i_1} \ldots \langle\top\rangle_{i_k}$. Suppose that α is of the form $\boxdot E$. Since $M \vDash \alpha$, by Lemma 4.1.2, we have $M', w \vDash E$. By Lemma 4.1.3, it follows that $\boxdot E \in Sat_L(I)$. Now suppose that α is of the form $\boxdot\Diamond_i E$. Since $M \vDash \alpha$, we have $M, w \vDash \Diamond_i E$, and by Lemma 4.1.2, $M', w \vDash \Diamond_i E$. There exists u such that $R_i(w, u)$ holds and $M', u \vDash E$. By Lemma 4.1.4, it follows that $\Diamond_i E \in H(w)$. Hence $\boxdot\Diamond_i E \in Sat_L(I)$ (by Lemma 4.1.3). •

Expected Lemma 3.3.4 *If P is an L-MProlog program then $P \vDash_L I_{L,P}$.*

Proof. Let $M = \langle D, W, \tau, R_1, \ldots, R_m, \pi\rangle$ be an arbitrary L-model of P and σ a *maximal* $\Diamond$-realization function on M (see Definition 3.1.3). It is straightforward to prove by induction on n that $M, \sigma \vDash T_{L,P}\uparrow n$. In fact, if $M, \sigma \vDash T_{L,P}\uparrow n$, then $M, \sigma \vDash Sat_L(T_{L,P}\uparrow n)$, and hence $M, \sigma \vDash T_{0L,P}(Sat_L(T_{L,P}\uparrow n))$. Since $NF_L(I) = I$ for any I, it follows that $M, \sigma \vDash T_{L,P}(T_{L,P}\uparrow n)$. Therefore $M, \sigma \vDash I_{L,P}$. •

Expected Lemma 3.3.5 *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.*

Proof. Let I' be the least extension of I such that, if $\boxdot\varphi$ is a program clause of P, $\varphi = (A \leftarrow B_1, \ldots, B_n)$, and ψ is a ground instance of φ, then $\boxdot p_\psi \in I'$, where p_ψ is a fresh 0-ary predicate symbol. Let M and M' be the extended L-model graphs of I and I', respectively. It is easy to see that these model graphs have the same frame. Let $M = \langle W, \tau, R_1, \ldots, R_m, H\rangle$ and $M' = \langle W, \tau, R_1, \ldots, R_m, H'\rangle$. Clearly, M is an L-model. By Lemma 4.1.2, it suffices to show that $M \vDash P$.

Let $\boxdot\varphi$ be a program clause of P, $\varphi = (A \leftarrow B_1, \ldots, B_n)$, and ψ a ground instance of φ. By Lemmas 3.2.2 and 4.1.2, $M' \vDash \boxdot p_\psi$. To prove that $M \vDash P$ it is sufficient to show that for any $w \in W$, if $p_\psi \in H'(w)$ then $M, w \vDash \psi$. Suppose that $p_\psi \in H'(w)$.

Let $\triangle = w$ and $\boxdot' = \Box_{i_1} \ldots \Box_{i_k}$ be a $\Box$-lifting form of $\triangle$. By Lemma 4.1.3, some $\Box$-lifting form of $\triangle p_\psi$ belongs to $Sat_L(I')$. This $\Box$-lifting form must be $\boxdot' p_\psi$. Thus $\boxdot' p_\psi \in Sat_L(\{\boxdot p_\psi\})$. Hence $\boxdot p_\psi \rightarrow \boxdot' p_\psi$ is L-valid and the program clause $\boxdot'\psi$ is a ground L-instance of $\boxdot\varphi$.

Let $\psi = (A' \leftarrow B'_1, \ldots, B'_n)$ and suppose that $M, w \vDash B'_i$ for all $1 \leq i \leq n$. We need to show that $M, w \vDash A'$. For this, we first show that a $\Box$-lifting form of $\triangle B'_i$ belongs to $Sat_L(I)$ for every $1 \leq i \leq n$. Consider the following cases:

- Case B'_i is a classical atom: The assertion follows from Lemma 4.1.3.
- Case B'_i is of the form $\Box_j E$: Since $M, w \vDash B'_i$, it follows that $M, w\langle\top\rangle_j \vDash E$, and by Lemma 4.1.3, some $\Box$-lifting form of $\triangle\langle\top\rangle_j E$ belongs to $Sat_L(I)$, which means that some $\Box$-lifting form of $\triangle B'_i$ belongs to $Sat_L(I)$.
- Case B'_i is of the form $\Diamond_j E$: Since $M, w \vDash B'_i$, there exists a world u such that $R_j(w, u)$ holds and $M, u \vDash E$. By Lemma 4.1.4, it follows that $\Diamond_j E \in H(w)$. Hence, by Lemma 4.1.3, some $\Box$-lifting form of $\triangle B'_i$ belongs to $Sat_L(I)$.

Therefore, by the definition of $T_{0L,P}$, some $\Box$-lifting form α of $\triangle A''$, where A'' is the forward labeled form of A', belongs to $T_{0L,P}(Sat_L(I))$. Since $T_{0L,P}(Sat_L(I)) = T_{L,P}(I) \subseteq I$, by Lemma 3.2.2, we have that $M, \sigma \vDash \alpha$, where σ is the standard $\Diamond$-realization function on M. Hence $M, w \vDash A'$. Thus $M, w \vDash \psi$, which completes the proof. •

Expected Lemma 3.5.1 *Let $\triangle$ and $\triangle'$ be ground modalities in L-normal labeled form. Let B be an atom of the form E, $\Diamond_i E$, or $\Box_i E$, and B' an atom of the form E, $\Diamond_j E$, $\langle X\rangle_j E$, or $\Box_j E$, where X is a fresh atom variable. Suppose that $\triangle$ is an L-instance of $\triangle'$ and B is an L-instance of B'. Then $\triangle' B'$ is derivable from $\triangle B$ using $rSat_L$.*

Proof. We have that $\triangle'$ is a $\Box$-lifting form of $\triangle$, and either B' is a $\Box$-lifting form of B or B' is of the form $\langle X\rangle_j$ and B is of the form $\Diamond_j$. Hence $\triangle' B'$ is derivable from $\triangle B$ using applications of the $rSat_L$ rules $\triangle\nabla_i\alpha \leftarrow \triangle\Box_i\alpha$ and $\triangle\Diamond_i\alpha \leftarrow \triangle\langle X\rangle_i\alpha$. •

Expected Lemma 3.5.2 *Suppose that β is an atom in almost L-normal labeled form and $\alpha \in Sat_L(\{\beta\})$ or $\alpha \in NF_L(\{\beta\})$. Then there exists an atom β' and a substitution θ s.t. $\beta = \beta'\theta$, the domain of θ consists of fresh atom variables, and β' is derivable from α using $rSat_L$ and rNF_L.*

Proof. Note that NF_L is the identity operator and we can ignore it. If α is derived from β using Sat_L rules identified by $(i_1), \ldots, (i_k)$, then by applying the sequence of $rSat_L$ rules identified

by $(i_k), \ldots, (i_1)$ to α we obtain an atom β' such that $\beta = \beta'\theta$, where θ is a substitution with domain consisting of fresh atom variables. •

Expected Lemma 3.5.3 *Let $\beta = rSat_L(\alpha)$, M be an L-model, σ a $\Diamond$-realization function on M, and θ a substitution. Suppose that $M, \sigma \vDash \forall_c(\beta'\theta)$ for some $\Box$-lifting form β' of β. Then $M, \sigma \vDash \forall_c(\alpha'\theta)$ for some $\Box$-lifting form α' of α.*

Proof. If the rule used to derive β from α is $\triangle\nabla_i\gamma \leftarrow \triangle\Box_i\gamma$, where γ denotes an atom, then just let $\alpha' = \beta'$. The remaining cases are similar to each other, and we consider, e.g., the case when the used rule is $\triangle\Box_j\Diamond_k\gamma \leftarrow \triangle\Diamond_i\gamma$. We have that $\alpha = \triangle\Box_j\Diamond_k\gamma$ and $\beta = \triangle\Diamond_i\gamma$. Let $\beta' = \triangle'\nabla_i\gamma'$. Since $M, \sigma \vDash \forall_c(\beta'\theta)$, we have $M, \sigma \vDash \forall_c(\triangle'\Diamond_i\gamma'\theta)$, and hence $M, \sigma \vDash \forall_c(\triangle'\Box_j\Diamond_k\gamma'\theta)$ (since $A5(i,j,k)$ holds). Choose $\alpha' = \triangle'\Box_j\Diamond_k\gamma'$. •

Expected Lemma 3.5.4 *Let $\beta =_\delta rNF_L(\alpha)$, M be an L-model, σ a maximal $\Diamond$-realization function on M, and θ a substitution. Suppose that $M, \sigma \vDash \forall_c(\beta'\theta)$ for some $\Box$-lifting form β' of β. Then $M, \sigma \vDash \forall_c(\alpha'\delta\theta)$ for some $\Box$-lifting form α' of α.*

Proof. This lemma is irrelevant for $L = BSMM$, because there are no rules specifying rNF_L. •

4.2 A Schema for Semantics of $sCFG$-MProlog

In this section, let L be an $sCFG$ logic. By $Axioms(L)$ we denote the set of axioms of L. In Table 4.2, we present a schema for semantics of $sCFG$-MProlog. By Corollary 2.5.2, the problem of checking whether $\boxdot$ is an L-context instance of $\boxdot'$ is decidable.

When resolving a goal with an input clause, if we relax the condition that "the mgu θ unifies the selected head atom A' with the forward labeled form A'' of the head of the input clause" by requiring only that θ is a most general substitution such that $A'\theta$ and $A''\theta$ have the same classical atom and $A'\theta$ is an L-instance of $A''\theta$, then the $rSat_L$ rule (6) can be discarded without violating soundness and completeness of the calculus. In fact, it can be shown that every SLD-refutation in the original calculus can be simulated in the new calculus by another one with a more general computed answer, and vice versa.

Observe that our SLD-resolution calculus for $sCFG$ is more specific and efficient than the calculus given for $BSMM$, as the problem of checking L-context instances is decidable for $sCFG$, while it is currently not known whether such a problem is decidable for the whole class $BSMM$. Furthermore, the order $\preceq_L$ between modal operators is defined more sophisticatedly for $sCFG$ than for $BSMM$.

Theorem 4.2.1 *The schema given in Table 4.2 for semantics of sCFG-MProlog is correct.*

To prove this theorem we have to prove Expected Theorem 3.2.3 and Expected Lemmas 3.2.2, 3.3.2, 3.3.4, 3.3.5, 3.5.1 – 3.5.4. (Expected Lemma 3.3.1 clearly holds.) To do this we also use *extended L-model graphs* as defined in Definition 4.1.1 (at page 48) for the case of $BSMM$. It is easy to see that the auxiliary Lemmas 4.1.2 and 4.1.3 still hold for $L = sCFG$.

Lemma 4.2.2 *If $\Box_i \preceq \Box_j$ or $\Diamond_j \preceq \Diamond_i$ then $\triangle\Diamond_i\alpha$ is derivable from $\triangle\Diamond_j\alpha$ using the $rSat_L$ rule (5).*

$L = sCFG$, L-MProlog	
$\preceq_L$ is defined by Definition 3.3.1 at page 34.	
No restrictions on L-normal form of modalities. No rules specifying NF_L and rNF_L.	
In what follows, (*) denotes $\Box_i\varphi \to \Box_{j_1} \dots \Box_{j_k}\varphi \in Axioms(L)$.	
Rules specifying	
Ext_L	$\triangle\Box_i\alpha \to \triangle\Box_{j_1} \dots \Box_{j_k}\alpha$ if (*) (1)
Sat_L	the rule specifying Ext_L plus $\triangle\nabla^{(1)} \dots \nabla^{(k)}\alpha \to \triangle\Diamond_i\alpha$ if (*) and $\Diamond_{j_t} \preceq_L \nabla^{(t)}$ for all $1 \le t \le k$ (2)
$rSat_L$	$\triangle\Diamond_i\alpha \leftarrow \triangle\langle X\rangle_i\alpha$ for X being a fresh atom variable (3) $\triangle\nabla^{(1)} \dots \nabla^{(k)}\alpha \leftarrow \triangle\Box_i\alpha$ if (*) and $\nabla^{(t)} \preceq_L \Box_{j_t}$ for all $1 \le t \le k$ (4) $\triangle\Diamond_i\alpha \leftarrow \triangle\Diamond_{j_1} \dots \Diamond_{j_k}\alpha$ if (*) (5) $\triangle\nabla\alpha \leftarrow \triangle\Box_i\alpha$ if $\nabla \preceq_L \Box_i$ (6)

Table 4.2: A schema for semantics of MProlog in $sCFG$ logics

Proof. Suppose that $\Box_i \preceq \Box_j$ or $\Diamond_j \preceq \Diamond_i$. Thus $\Box_j\varphi \to \Box_i\varphi$ is L-valid (for every φ). By Proposition 2.5.1, i is derivable from j using the context-free grammar $\mathcal{G}(L)$. This derivation corresponds to a sequence of implications $\Box_j\varphi \to \dots \to \Box_i\varphi$, where each implication is of the form $\triangle'\Box_s\triangle''\varphi \to \triangle'\Box_{t_1} \dots \Box_{t_k}\triangle''\varphi$ with $\Box_s\psi \to \Box_{t_1} \dots \Box_{t_k}\psi$ being an axiom of L. The sequence of dual implications is $\Diamond_j\varphi \leftarrow \dots \leftarrow \Diamond_i\varphi$, where each implication is of the form $\triangle'\Diamond_s\triangle''\varphi \leftarrow \triangle'\Diamond_{t_1} \dots \Diamond_{t_k}\triangle''\varphi$ with $\Box_s\psi \to \Box_{t_1} \dots \Box_{t_k}\psi$ being an axiom of L. This implies that $\triangle\Diamond_i\alpha$ is derivable from $\triangle\Diamond_j\alpha$ using the $rSat_L$ rule (5). •

The following lemma is very similar to Lemma 4.1.4 of the case of $BSMM$.

Lemma 4.2.3 *Let I be a model generator and $M = \langle W, \tau, R_1, \dots, R_m, H\rangle$ be the extended L-model graph of I. Then for any w and u such that $R_i(w,u)$ holds: i) if $\Box_i\alpha \in H(w)$ then $\alpha \in H(u)$, ii) if $\alpha \in H(u)$ then there exists a modal operator ∇ such that $\nabla\alpha \in H(w)$ and $\Diamond_i \preceq_L \nabla$.*

Proof. Let $(R'_j)_{1\le j\le m}$ be the skeleton of M. We prove this lemma by induction on the number of steps needed to obtain $R_i(w,u)$ when extending $(R'_j)_{1\le j\le m}$ to $(R_j)_{1\le j\le m}$.

Since $R_i(w,u)$ holds, there are two cases to consider:

- Case $u = w\langle E\rangle_i$ and $R'_i(w, w\langle E\rangle_i)$: The assertions hold by the definition of M.
- Case $\Box_i\varphi \to \Box_{j_1} \dots \Box_{j_k}\varphi$ is an axiom of L and $R_i(w,u)$ is created from $R_{j_1}(w_0,w_1), \dots, R_{j_k}(w_{k-1},w_k)$, where $w = w_0$ and $u = w_k$:

Consider the first assertion. Suppose that $\Box_i\alpha \in H(w)$. By Lemma 4.1.3 for $L = sCFG$, there exists a $\Box$-lifting form $\triangle$ of w such that $\triangle\Box_i\alpha \in Ext'_L(I)$. Since $\triangle\Box_i\alpha \in Ext'_L(I)$, we have $\triangle\Box_{j_1}\ldots\Box_{j_k}\alpha \in Ext'_L(I)$, and by Lemma 4.1.3 for $L = sCFG$, $\Box_{j_1}\ldots\Box_{j_k}\alpha \in H(w)$. Hence, by the inductive assumption, we can derive that $\alpha \in H(u)$.

Consider the second assertion. Suppose that $\alpha \in H(u)$. For each $t = k, (k-1), \ldots, 1$, by the inductive assumption, there exists a modal operator $\nabla^{(t)}$ such that $\nabla^{(t)}\nabla^{(t+1)}\ldots\nabla^{(k)}\alpha \in H(w_{t-1})$ and $\Diamond_{j_t} \preceq_L \nabla^{(t)}$. Thus, we have $\nabla^{(1)}\ldots\nabla^{(k)}\alpha \in H(w)$, where $\Diamond_{j_t} \preceq_L \nabla^{(t)}$ for every $1 \leq t \leq k$. By Lemma 4.1.3 for $L = sCFG$, there exists a $\Box$-lifting form $\triangle$ of w such that $\triangle\nabla^{(1)}\ldots\nabla^{(k)}\alpha \in Ext'_L(I)$. By the definition of Ext'_L, it follows that $\triangle\Diamond_i\alpha \in Ext'_L(I)$. Hence, by Lemma 4.1.3 for $L = sCFG$, $\Diamond_i\alpha \in H(w)$.

•

The proofs of Expected Lemmas 3.2.2, 3.3.2, 3.3.5 for the case of $BSMM$ are appliable for the case $L = sCFG$ when Lemma 4.2.3 is used instead of Lemma 4.1.4. For example, to use the proof of Expected Lemma 3.3.5 of the case of $BSMM$ for the case $L = sCFG$ we only need to change the text

By Lemma 4.1.4, it follows that $\Diamond_j E \in H(w)$. Hence, by Lemma 4.1.3, some $\Box$-lifting form of $\triangle B'_i$ belongs to $Sat_L(I)$.

by

By Lemma 4.2.3, there exists a modal operator ∇ such that $\nabla E \in H(w)$ and $\Diamond_j \preceq_L \nabla$. By Lemma 4.1.3 for $L = sCFG$, some $\Box$-lifting form of $\triangle\nabla E$ belongs to $Sat_L(I)$. It follows that $\triangle B'_i$ is an L-instance of some atom of $Sat_L(I)$.

The proofs of Expected Theorem 3.2.3 and Expected Lemma 3.3.4 for the case of $BSMM$ are applicable for the case $L = sCFG$ with no changes. Expected Lemma 3.5.4 is irrelevant for $L = sCFG$ because there are no rules specifying rNF_L. So, there remain to prove only Expected Lemmas 3.5.1 – 3.5.3.

Expected Lemma 3.5.1 *Let $\triangle$ and $\triangle'$ be ground modalities in L-normal labeled form. Let B be an atom of the form E, $\Diamond_i E$, or $\Box_i E$, and B' an atom of the form E, $\Diamond_j E$, $\langle X\rangle_j E$, or $\Box_j E$, where X is a fresh atom variable. Suppose that $\triangle$ is an L-instance of $\triangle'$ and B is an L-instance of B'. Then $\triangle' B'$ is derivable from $\triangle B$ using $rSat_L$.*

Proof. $\triangle' B$ is derivable from $\triangle B$ using the $rSat_L$ rule (6). If B' is of the form $\Diamond_j E$ then B must be of the form $\Diamond_i E$ with $\Diamond_i \preceq_L \Diamond_j$, and by Lemma 4.2.2, $\triangle' B'$ is derivable from $\triangle' B$ using the $rSat_L$ rule (5). Similarly, if B' is of the form $\langle X\rangle_j E$ then B must be of the form $\Diamond_i E$ with $\Diamond_i \preceq_L \Diamond_j$, and $\triangle' B'$ is derivable from $\triangle' B$ using the $rSat_L$ rules (5) and (3). If B' is of the form $\Box_j E$ then $\triangle' B'$ is derivable from $\triangle' B$ using the $rSat_L$ rule (6). In the remaining case, B' and B are equal to a classical atom. We conclude that $\triangle' B'$ is derivable from $\triangle B$ using $rSat_L$. •

Expected Lemma 3.5.2 *Suppose that β is an atom in almost L-normal labeled form and $\alpha \in Sat_L(\{\beta\})$ or $\alpha \in NF_L(\{\beta\})$. Then there exists an atom β' and a substitution θ s.t. $\beta = \beta'\theta$, the domain of θ consists of fresh atom variables, and β' is derivable from α using $rSat_L$ and rNF_L.*

Proof. Note that NF_L is the identity operator and we can ignore it. It is sufficient to consider the case when α is derived from β using an application of one Sat_L rule. If the used rule is

(1) then β is derivable from α using the $rSat_L$ rule (4) and we can choose $\beta' = \beta$ and $\theta = \varepsilon$. Suppose that the Sat_L rule used to derive α from β is (2). Let $\beta = \triangle\nabla_{i_1}^{(1)} \dots \nabla_{i_k}^{(k)}\gamma$ and $\alpha = \triangle\Diamond_i\gamma$, where $\Box_i\varphi \to \Box_{j_1} \dots \Box_{j_k}\varphi$ is an axiom of L and $\Diamond_{j_t} \preceq_L \nabla_{i_t}^{(t)}$ for all $1 \leq t \leq k$. We have that $\Diamond_{j_t} \preceq_L \Diamond_{i_t}$ for all $1 \leq t \leq k$. The atom $\triangle\Diamond_{j_1} \dots \Diamond_{j_k}\gamma$ is derivable from $\triangle\Diamond_i\gamma$ using the $rSat_L$ rule (5). By Lemma 4.2.2, the atom $\triangle\Diamond_{i_1} \dots \Diamond_{i_k}\gamma$ is derivable from $\triangle\Diamond_{j_1} \dots \Diamond_{j_k}\gamma$ using the $rSat_L$ rule (5). Finally, using the $rSat_L$ rules (3) and (6), we can derive from $\triangle\Diamond_{i_1} \dots \Diamond_{i_k}\gamma$ an atom $\beta' = \triangle\nabla_{i_1}^{(1')} \dots \nabla_{i_k}^{(k')}\gamma$ such that, for every $1 \leq t \leq k$, $\nabla_{i_t}^{(t')} = \Box_{i_t}$ if $\nabla_{i_t}^{(t)} = \Box_{i_t}$, and $\nabla_{i_t}^{(t')} = \langle X_t\rangle_{i_t}$ for some fresh atom variable X_t if $\nabla_{i_t}^{(t)} = \langle E_t\rangle_{i_t}$. Thus β' is derivable from α using $rSat_L$ and we have that $\beta = \beta'\theta$ for some substitution θ with domain consisting of fresh atom variables. •

Expected Lemma 3.5.3 *Let $\beta = rSat_L(\alpha)$, M be an L-model, σ a $\Diamond$-realization function on M, and θ a substitution. Suppose that $M, \sigma \vDash \forall_c(\beta'\theta)$ for some $\Box$-lifting form β' of β. Then $M, \sigma \vDash \forall_c(\alpha'\theta)$ for some $\Box$-lifting form α' of α.*

Proof. There are the following cases to consider:

- Case when the rule (3) is used, $\alpha = \triangle\Diamond_i E$ and $\beta = \triangle\langle X\rangle_i E$: Let $\beta' = \triangle'\nabla_i E$ and choose $\alpha' = \triangle'\Diamond_i E$.

- Case when the rule (4) is used, $\alpha = \triangle\nabla_{i_1}^{(1)} \dots \nabla_{i_k}^{(k)}\gamma$ and $\beta = \triangle\Box_i\gamma$, where $\nabla_{i_t}^{(t)} \preceq_L \Box_{j_t}$ for $1 \leq t \leq k$ and $\Box_i\varphi \to \Box_{j_1} \dots \Box_{j_k}\varphi$ is an axiom of L : Let $\beta' = \triangle'\Box_i\gamma'$ and choose $\alpha' = \triangle'\Box_{i_1} \dots \Box_{i_k}\gamma'$.

- Case when the rule (5) is used, $\alpha = \triangle\Diamond_i\gamma$ and $\beta = \triangle\Diamond_{j_1} \dots \Diamond_{j_k}\gamma$, where $\Box_i\varphi \to \Box_{j_1} \dots \Box_{j_k}\varphi$ is an axiom of L : Let $\beta' = \triangle'\nabla_{j_1}^{(1)} \dots \nabla_{j_k}^{(k)}\gamma'$ and choose $\alpha' = \triangle'\Diamond_i\gamma'$.

- Case when the rule (6) is used, $\alpha = \triangle\nabla_j\gamma$ and $\beta = \triangle\Box_i\gamma$, where $\nabla_j \preceq_L \Box_i$: Let $\beta' = \triangle'\Box_i\gamma'$ and choose $\alpha' = \triangle'\Box_j\gamma'$.

Note that α' is a $\Box$-lifting form of α. Since $M, \sigma \vDash \forall_c(\beta'\theta)$, it is easy to see that $M, \sigma \vDash \forall_c(\alpha'\theta)$. •

4.3 Programming about Multi-degree Belief

Our SLD-resolution calculus for MProlog in BSMM is elegant like a Hilbert-style axiom system, but similarly to using a Hilbert-style axiom system for automatic reasoning, it is not very efficient. The calculus may be too "syntactic". For more specific modal logics like the considered multimodal logics of belief, we want to have more efficient SLD-resolution calculi. For this aim, we look more deeply at "semantical" properties of the considered logics and use advanced techniques introduced for our framework like normalizing modalities or ordering modal operators.

To reason about multi-degree belief we can use the multimodal logics $KDI4$, $KDI4_s$, $KDI4_s5$, and $KDI45$. Recall that, in these logics, $\Box_i\varphi$ stands for "φ is believed up to degree i" and $\Diamond_i\varphi$ stands for "it is possible weakly at degree i that φ". In this section, we present schemata for semantics of MProlog in the mentioned logics and prove their correctness.

$L = KDI4_s$, L-MProlog	
$\preceq_L$ is defined by Definition 3.3.1.	
No restrictions on L-normal form of modalities. No rules specifying NF_L and rNF_L.	
Rules specifying	
Ext_L	$\triangle\Box_i\alpha \to \triangle\Box_j\alpha$ if $i > j$ (1) $\triangle\Box_i\alpha \to \triangle\Box_j\Box_i\alpha$ (2)
Sat_L	the rules specifying Ext_L plus $\triangle\nabla\nabla' E \to \triangle\Diamond_i E$ if $\Diamond_i \preceq_L \nabla'$ (3)
$rSat_L$	$\triangle\Diamond_i E \leftarrow \triangle\langle X\rangle_i E$ for X being a fresh atom variable (4) $\triangle\nabla_i\alpha \leftarrow \triangle\Box_j\alpha$ if $i \leq j$ (5) $\triangle\Diamond_i E \leftarrow \triangle\Diamond_j E$ if $i > j$ (6) $\triangle\nabla\Box_i\alpha \leftarrow \triangle\Box_i\alpha$ (7) $\triangle\Diamond_i E \leftarrow \triangle\langle X\rangle_j\Diamond_i E$ for X being a fresh atom variable (8)

Table 4.3: A schema for semantics of $KDI4_s$-MProlog

4.3.1 Schemata for Semantics of MProlog in $KDI4_s$ and $KDI4$

Even though $KDI4_s$ and $KDI4$ belong to both of the classes $BSMM$ and $sCFG$, they are specific modal logics, for which we can develop more efficient schemata for semantics of MProlog. In this subsection, L denotes $KDI4_s$ or $KDI4$.

In Table 4.3, we give a schema for semantics of $KDI4_s$-MProlog. The rule (1) in that schema follows from the axiom (I); the rules (2) and (7) are based on the axiom (4_s); the rules (3) and (8) are based on the reverse of the axiom (4_s), i.e. $\Diamond_j\Diamond_i\varphi \to \Diamond_i\varphi$; the rule (5) is based on the axioms (D) and (I); and finally, the rule (6) follows from the reverse of the axiom (I), i.e. $\Diamond_j\varphi \to \Diamond_i\varphi$ if $i > j$.

In Table 4.4, we give a schema for semantics of $KDI4$-MProlog. The rule (1) in that schema follows from the axiom (I); the rules (2) and (7) are based on the axiom (4); the rules (3) and (8) are based on the reverse of the axiom (4), i.e. $\Diamond_i\Diamond_i\varphi \to \Diamond_i\varphi$; the rule (5) is based on the axioms (D) and (I); and finally, the rule (6) follows from the reverse of the axiom (I).

For $L \in \{KDI4_s, KDI4\}$, a universal modality $\boxdot$ is an L-context instance of $\Box_i$ iff:

- $L = KDI4_s$ and the last modal operator of $\boxdot$ is $\Box_j$ with $j \leq i$; or
- $L = KDI4$, $\boxdot$ is not empty, and every modal operator $\Box_j$ of $\boxdot$ satisfies $j \leq i$.

Theorem 4.3.1 *The schemata given in Tables 4.3 and 4.4 for semantics of MProlog in $KDI4_s$ and $KDI4$ are correct.*

To prove this theorem we have to prove Expected Theorem 3.2.3 and Expected Lemmas 3.2.2, 3.3.1, 3.3.2, 3.3.4, 3.3.5, 3.5.1 – 3.5.4.

$L = KDI4$, L-MProlog		
$\preceq_L$ is defined by Definition 3.3.1.		
No restrictions on L-normal form of modalities. No rules specifying NF_L and rNF_L.		
Rules specifying		
Ext_L	$\triangle\Box_i\alpha \rightarrow \triangle\Box_j\alpha$ if $i > j$	(1)
	$\triangle\Box_i\alpha \rightarrow \triangle\Box_i\Box_i\alpha$	(2)
Sat_L	the rules specifying Ext_L plus	
	$\triangle\nabla\nabla' E \rightarrow \triangle\Diamond_i E$ if $\Diamond_i \preceq_L \nabla$ and $\Diamond_i \preceq_L \nabla'$	(3)
$rSat_L$	$\triangle\Diamond_i E \leftarrow \triangle\langle X\rangle_i E$ for X being a fresh atom variable	(4)
	$\triangle\nabla_i\alpha \leftarrow \triangle\Box_j\alpha$ if $i \leq j$	(5)
	$\triangle\Diamond_i E \leftarrow \triangle\Diamond_j E$ if $i > j$	(6)
	$\triangle\Box_i\Box_i\alpha \leftarrow \triangle\Box_i\alpha$	(7)
	$\triangle\Diamond_i E \leftarrow \triangle\langle X\rangle_j\Diamond_i E$ for X being a fresh atom variable and $i \geq j$	(8)

Table 4.4: A schema for semantics of $KDI4$-MProlog

Expected Lemmas 3.3.1 (page 35) and 3.5.1 – 3.5.4 (page 40) and can easily be verified. The proofs of Expected Theorem 3.2.3 and Expected Lemma 3.3.4 are similar to the ones given for $BSMM$ (at pages 50 and 51, respectively).

Expected Lemma 3.2.2 *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 $M, \sigma \vDash I$.*

Proof. By the definition, M is an L-model. Let $(R'_i)_{1\leq i\leq m}$ be the skeleton of M. We prove by induction on the length of α that for any $w \in W$, if $\alpha \in H(w)$, then $M, \sigma, w \vDash \alpha$. Let $w = \langle E_1\rangle_{i_1} \ldots \langle E_k\rangle_{i_k}$ and $\triangle = w$ (be a modality).

The cases when α is a classical atom or $\alpha = \langle E\rangle_i\beta$ are trivial. Suppose that $\alpha = \Box_i\beta$. Let u be a world such that $R_i(w, u)$ holds. By the inductive assumption, it suffices to show that $\beta \in H(u)$. Since $R_i(w, u)$ holds, there are two cases:

- Case $L = KDI4_s$, $w_0 = w$, $R'_{j_1}(w_0, w_1)$, $R'_{j_2}(w_1, w_2)$, $\ldots$, $R'_{j_h}(w_{h-1}, w_h)$, $R'_j(w_h, u)$, $h \geq 0$ and $j \leq i$: Since $\Box_i\beta \in H(w)$, by Lemma 3.2.1, there exists a $\Box$-lifting form $\triangle'$ of $\triangle$ such that $\triangle'\Box_i\beta \in Ext_L(I)$. By the definition of $Ext_L(I)$, it follows that $\triangle'\Box_{j_1} \ldots \Box_{j_h}\Box_j\beta \in Ext_L(I)$. Hence, by Lemma 3.2.1, $\beta \in H(u)$.

- Case $L = KDI4$, $w_0 = w$, $R'_{j_1}(w_0, w_1)$, $R'_{j_2}(w_1, w_2)$, $\ldots$, $R'_{j_h}(w_{h-1}, w_h)$, $R'_j(w_h, u)$, $h \geq 0$ and $k \leq i$ for all $k \in \{j_1, \ldots, j_h, j\}$: The assertion holds by the same argumentation as for the above case.

•

Expected Lemma 3.3.2 *Let I be an L-normal model generator, M the standard L-model of I, and α a ground L-MProlog goal atom. Suppose that $M \vDash \alpha$. Then α is an L-instance of some atom of $Sat_L(I)$.*

Proof. If α is of the form $\boxdot E$ then $\alpha \in Ext_L(I)$ (since $M \vDash \alpha$), and hence $\alpha \in Sat_L(I)$. Suppose that $\alpha = \boxdot\Diamond_i E$, where $\boxdot = \Box_{i_1} \ldots \Box_{i_k}$ with $k \geq 0$. Let $\langle W, \tau, R_1, \ldots, R_m, H\rangle$ be the standard L-model graph of I and $\triangle = w = \langle\top\rangle_{i_1} \ldots \langle\top\rangle_{i_k}$. Since $M \vDash \alpha$, we have $M, w \vDash \Diamond_i E$. There are two cases:

- Case $L = KDI4_s$, $E \in H(u)$, $u = w\langle F_1\rangle_{j_1} \ldots \langle F_h\rangle_{j_h}\langle F\rangle_j$, $h \geq 0$, $j \leq i$: By Lemma 3.2.1, some $\Box$-lifting form of $\triangle\langle F_1\rangle_{j_1} \ldots \langle F_h\rangle_{j_h}\langle F\rangle_j E$ belongs to $Ext_L(I)$. It follows that some $\Box$-lifting form of $\boxdot\langle F_1\rangle_{j_1} \ldots \langle F_h\rangle_{j_h}\langle F\rangle_j E$ belongs to $Sat_L(I)$. Hence $\boxdot\Diamond_i E \in Sat_L(I)$.

- Case $L = KDI4$, $E \in H(u)$, $u = w\langle F_1\rangle_{j_1} \ldots \langle F_h\rangle_{j_h}\langle F\rangle_j$, $h \geq 0$, $j_1 \leq i$, $\ldots$, $j_h \leq i$, $j \leq i$: The assertion holds by a similar argumentation as for the above case.

•

Expected Lemma 3.3.5 *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.*

Proof. Let M be the standard L-model of I and σ the standard $\Diamond$-realization function on M. By the definition of L-instances of program clauses and the construction of M, it is sufficient to prove that for any ground L-instance $\boxdot(A \leftarrow B_1, \ldots, B_n)$ of some program clause of P, for any $w \in W$ being an L-instance of $\boxdot$, $M, w \vDash (A \leftarrow B_1, \ldots, B_n)$. Suppose that $M, w \vDash B_i$ for all $1 \leq i \leq n$. We show that $M, w \vDash A$.

Let $\triangle = w$. We first show that for any ground simple atom B of the form E, $\Box_i E$, or $\Diamond_i E$, if $M, w \vDash B$ then $\triangle B$ is an L-instance of some atom from $Sat_L(I)$. If B is of the form E or $\Box_i E$, then by Lemma 3.2.1, some $\Box$-lifting form of $\triangle B$ belongs to $Ext_L(I)$, and hence $\triangle B$ is an L-instance of some atom from $Sat_L(I)$. Now consider the case when B is of the form $\Diamond_i E$. There exists u such that $R_i(w, u)$ and $M, u \vDash E$. There are two cases:

- Case $L = KDI4_s$, $u = w\langle F_1\rangle_{j_1} \ldots \langle F_h\rangle_{j_h}\langle F\rangle_j$, $h \geq 0$, $j \leq i$: By Lemma 3.2.1, some $\Box$-lifting form of $\triangle\langle F_1\rangle_{j_1} \ldots \langle F_h\rangle_{j_h}\langle F\rangle_j E$ belongs to $Ext_L(I)$. It follows that $\triangle\langle F_1\rangle_{j_1} \ldots \langle F_h\rangle_{j_h}\langle F\rangle_j E$ is an L-instance of some atom from $Sat_L(I)$, and hence $\triangle\Diamond_i E$ is also an L-instance of some atom from $Sat_L(I)$.

- Case $L = KDI4$, $u = w\langle F_1\rangle_{j_1} \ldots \langle F_h\rangle_{j_h}\langle F\rangle_j$, $h \geq 0$, $j_1 \leq i$, $\ldots$, $j_h \leq i$, $j \leq i$: The assertion holds by a similar argumentation as for the above case.

Since $M, w \vDash B_i$ for $1 \leq i \leq n$, it follows that $\triangle B_i$ is an L-instance of some atom from $Sat_L(I)$. Consequently, $\triangle A$ is an L-instance of some atom α from $T_{0L,P}(Sat_L(I))$. Since $T_{0L,P}(Sat_L(I)) = T_{L,P}(I) \subseteq I$, by Lemma 3.2.2, we have that $M, \sigma \vDash \alpha$, and hence $M, w \vDash A$.

•

4.3.2 A Schema for Semantics of $KDI4_s5$-MProlog

In this subsection, L denotes the logic $KDI4_s5$. 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 $1 \leq i \leq m$. (Recall that m is the maximal

$L = KDI4_s5$, L-MProlog	
$\preceq_L$ is defined by Definition 3.3.1 at page 34.	
A modality is in L-normal form if its length ≤ 1.	
Rules specifying	
Ext_L	$\Box_i E \rightarrow \Box_j E$ if $i > j$ (1)
Sat_L	the rules specifying Ext_L plus
	$\Box_i E \rightarrow \Box_m \Box_i E$ (2)
	$\langle F \rangle_i E \rightarrow \Box_m \Diamond_i E$ (3)
NF_L	$\nabla\nabla' E \rightarrow \nabla' E$ if ∇' is of the form $\Box_i$ or $\langle E \rangle_i$ (4)
rNF_L	$\nabla E \leftarrow \langle X \rangle_j \nabla E$ if ∇ is of the form $\Box_i$ or $\langle E \rangle_i$ and X is a fresh atom variable (5)
$rSat_L$	$\triangle\Diamond_i E \leftarrow \triangle\langle X \rangle_i E$ for X being a fresh atom variable (6)
	$\triangle\nabla_i \alpha \leftarrow \triangle\Box_j \alpha$ if $i \leq j$ (7)
	$\triangle\Diamond_i E \leftarrow \triangle\Diamond_j E$ if $i > j$ (8)
	$\nabla\nabla' E \leftarrow \nabla' E$ if ∇' is of the form $\Box_i$ or $\Diamond_i$ (9)

Table 4.5: A schema for semantics of $KDI4_s5$-MProlog

modal index; and we use E to denote a classical atom, A to denote a simple atom of the form E or ∇E, where ∇ is a modal operator, and α to denote an atom of the form $\triangle E$.)

In Section 3 we have given several small examples involving with $KDI4_s5$. In Table 4.5, we present a full schema for semantics of $KDI4_s5$-MProlog. L-normal form of modalities and the rules (2)–(5) and (9) in that schema are justified by the L-tautology $\nabla\varphi \equiv \nabla'\nabla\varphi$ with ∇ and ∇' being unlabeled modal operators. The rule (1) follows from the axiom (I), the rule (7) is based on the axioms (D) and (I), and the rule (8) follows from the reverse of the axiom (I). Note that $\boxdot$ is an L-context instance of $\boxdot'$ iff $\boxdot$ is an L-instance of $\boxdot'$ (here, $|\boxdot| = |\boxdot'| = 1$).

The schema given in Table 4.5 is formulated so that it can use the proofs given in Section 3. However, the rules (6), (7), (8) of Table 4.5 can be simplified by deleting the occurrences of $\triangle$ and replacing α by E without violating soundness and completeness of SLD-resolution. Furthermore, the rule (7) can be deleted if: a) the condition of the rule (5) that ∇ is of the form $\Box_i$ or $\langle E \rangle_i$ is deleted, b) when resolving a goal with an input clause, we relax the condition that "the mgu θ unifies the selected head atom A' with the forward labeled form A'' of the head of the input clause" by requiring only that θ is a most general substitution such that $A'\theta$ and $A''\theta$ have the same classical atom and $A'\theta$ is an L-instance of $A''\theta$. It can be shown that every SLD-refutation in the original calculus can be simulated in the new calculus by another one with a more general computed answer, and vice versa. This means that the new SLD-resolution calculus is sound and complete, provided that so is the original calculus.

Example 4.3.1 Reconsider the MProlog program P_{mdb} given in Example 2.9.1. To increase readability, we recall some clauses of P_{mdb}:

$$\varphi_5 = \Box_2(\Diamond_2 good\_in\_maths(x) \leftarrow good\_in\_physics(x))$$
$$\varphi_9 = \Box_5 physics\_student(Mike) \leftarrow$$

Here is an SLD-refutation of $P_{mdb} \cup \{\leftarrow \Diamond_2 good\_in\_maths(x)\}$ in $KDI4_s5$:

Goals	Input clauses/rules	MGUs, constraints
$\leftarrow \Diamond_2 good\_in\_maths(x)$		
$\leftarrow \langle X\rangle_2 good\_in\_maths(x)$	(6)	ε
$\leftarrow \langle Y\rangle_j \langle good\_in\_maths(x)\rangle_2 good\_in\_maths(x)$	(5)	$\{X/good\_in\_maths(x)\}$
$\leftarrow \langle Y\rangle_j good\_in\_physics(x)$	φ_5	$\{x_3/x\}, j \leq 2$
$\leftarrow \Box_j good\_in\_physics(x)$	(7)	$\varepsilon, j \leq 2$
$\diamond$	φ_9	$\{x/Mike\}$

The computed answer is $\{x/Mike\}$. In the above refutation, j can take value 1 or 2. In another work, we have implemented MProlog as an additional module to Prolog, and constraints as goal atoms. With that module, we can also consider, for example, the goals $\leftarrow \Box_i good\_in\_maths(x)$ and $\leftarrow \Diamond_i good\_in\_maths(x)$.

Theorem 4.3.2 *The schema given in Table 4.5 for semantics of $KDI4_s5$-MProlog is correct.*

To prove this theorem we have to prove Expected Theorem 3.2.3 and Expected Lemmas 3.2.2, 3.3.2, 3.3.4, 3.3.5, 3.5.1 – 3.5.4. (Expected Lemma 3.3.1 clearly holds.)

Expected Lemma 3.2.2 *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 $M, \sigma \vDash I$.*

Proof. By the definition, M is an L-model. Let $(R'_i)_{1\leq i\leq m}$ be the skeleton of M. We prove by induction on the length of α that for any $w \in W$, if $\alpha \in H(w)$, then $M, \sigma, w \vDash \alpha$. The cases when α is a classical atom or $\alpha = \langle E\rangle_i F$ (and $w = \tau$) are trivial. Consider the remaining case when $\alpha = \Box_i E$ and $w = \tau$. Let u be a world such that $R_i(\tau, u)$ holds. We show that $E \in H(u)$. Since $R_i(\tau, u)$, u must be of the form $\langle F\rangle_j$ for some F and $j \leq i$. Since $\Box_i E \in H(\tau)$, by the definition of Ext_L, we have $\Box_j E \in H(\tau)$, and hence $E \in H(u)$. •

Expected Theorem 3.2.3 *The standard L-model of an L-normal model generator I is a least L-model of I.*

Proof. Let $M = \langle W, \tau, R_1, \ldots, R_m, H\rangle$ be the standard L-model graph of I, σ the standard $\Diamond$-realization function and $(R'_i)_{1\leq i\leq m}$ the skeleton of the standard L-model of I. By Lemma 3.2.2, M is an L-model of I. Let $N = \langle D, W_2, \tau_2, S_1, \ldots, S_m, \pi\rangle$ be an arbitrary L-model of I and σ_2 a $\Diamond$-realization function on N such that $N, \sigma_2 \vDash I \cup Serial_L$.

We first show that if $R'_i(\tau, \langle E\rangle_i)$ holds then $\sigma_2(\tau_2, \langle E\rangle_i)$ is defined. The case $E = \top$ is trivial. Suppose that $R'_i(\tau, \langle E\rangle_i)$ holds and $E \neq \top$. Thus, there exists $\langle E\rangle_i \alpha \in H(\tau)$ for some α. Hence $N, \sigma_2, \tau_2 \vDash \langle E\rangle_i \alpha$, and $\sigma_2(w_2, \langle E\rangle_i)$ is defined.

Let $r \subseteq W \times W_2$ be the least relation such that, for all w, w_2, u_2, E, i:

- $r(\tau, \tau_2)$;
- if $R'_i(\tau, \langle E\rangle_i)$ holds then $r(\langle E\rangle_i, \sigma_2(\tau_2, \langle E\rangle_i))$;

- if $r(w, w_2)$ and $S_i(w_2, u_2)$ hold, then $r(\langle\top\rangle_i, u_2)$.

We prove that $M \leq_r N$. If $r(w, w_2)$ and $S_i(w_2, u_2)$ hold, then for $u = \langle\top\rangle_i$ we have $r(u, u_2)$ and $R_i(w, u)$. We proceed by showing that if $r(u, u_2)$ and $\alpha \in H(u)$ then $N, \sigma_2, u_2 \vDash \alpha$. The case $u = \tau$ is trivial. Suppose that $u = \langle E\rangle_i$, $r(u, u_2)$, and $\alpha \in H(u)$. There are two cases:

- $u_2 = \sigma_2(\tau_2, \langle E\rangle_i)$ and $R'_i(\tau, \langle E\rangle_i)$; or
- $E = \top$, $r(w, w_2)$, and $S_i(w_2, u_2)$, for some w, w_2.

Consider the first case. Since $\alpha \in H(u)$, either $\Box_i\alpha \in H(\tau)$ or $\langle E\rangle_i\alpha \in H(\tau)$. Hence, $N, \sigma_2, \tau_2 \vDash \Box_i\alpha$ or $N, \sigma_2, \tau_2 \vDash \langle E\rangle_i\alpha$. It follows that $N, \sigma_2, u_2 \vDash \alpha$.

Consider the second case. Since $\alpha \in H(u)$, it follows that $\Box_i\alpha \in H(\tau)$. Hence, $N, \sigma_2, \tau_2 \vDash \Box_i\alpha$. Since $r(w, w_2)$ and $S_i(w_2, u_2)$, it can be shown that u_2 is directly or indirectly reachable from τ_2 (via the accessibility relations S_j, $1 \leq j \leq m$). Hence $S_i(\tau_2, u_2)$ holds, and $N, \sigma_2, u_2 \vDash \alpha$.

To prove $M \leq_r N$, it remains to show that if $r(w, w_2)$ and $R_i(w, u)$ hold, then there exists $u_2 \in W_2$ such that $r(u, u_2)$ and $S_i(w_2, u_2)$ hold. Suppose that $r(w, w_2)$ and $R_i(w, u)$ hold. It follows that $R'_j(\tau, u)$ holds for some $j \leq i$. Let $u = \langle E\rangle_j$ and choose $u_2 = \sigma_2(\tau_2, \langle E\rangle_j)$. Thus we have $r(u, u_2)$. Since $r(w, w_2)$, it can be shown that w_2 is directly or indirectly reachable from τ_2 (via the accessibility relations S_k, $1 \leq k \leq m$). Hence $S_i(w_2, u_2)$ holds. •

Expected Lemma 3.3.1 *If $\Box_{i_1} \ldots \Box_{i_h}$ is a $\Box$-lifting form of a modality $\triangle$ in L-normal labeled form and $\triangle$ is an L-instance of $\boxdot$, then $\boxdot\varphi \vDash_L \Box_{i_1} \ldots \Box_{i_h}\varphi$ for any formula φ without labeled modal operators.*

Proof. Just note that $h = 1$ (since $\triangle$ is in L-normal labeled form) and $\Box_{i_1}$ is an L-instance of $\boxdot$. •

Expected Lemma 3.3.2 *Let I be an L-normal model generator, M the standard L-model of I, and α a ground L-MProlog goal atom. Suppose that $M \vDash \alpha$. Then α is an L-instance of some atom of $Sat_L(I)$.*

Proof. If α is of the form E or $\Box_i E$, then $\alpha \in Ext_L(I)$ (since $M \vDash \alpha$), and hence $\alpha \in Sat_L(I)$. Suppose that $\alpha = \Diamond_i E$. Let $\langle W, \tau, R_1, \ldots, R_m, H\rangle$ be the standard L-model graph of I. Since $M \vDash \alpha$, there exists a world $u = \langle F\rangle_j$ of M such that $j \leq i$ and $E \in H(u)$. By Lemma 3.2.1, some $\Box$-lifting form of $\langle F\rangle_j E$ belongs to $Ext_L(I)$. It follows that either $\Box_j E$ or $\langle F\rangle_j E$ belongs to $Ext_L(I)$. Hence α is an L-instance of some atom from $Sat_L(I)$. •

Expected Lemma 3.3.4 *If P is an L-MProlog program then $P \vDash_L I_{L,P}$.*

Proof. Let $M = \langle D, W, \tau, R_1, \ldots, R_m, \pi\rangle$ be an arbitrary L-model of P and σ a maximal $\Diamond$-realization function on M. Note that if $M, \sigma \vDash \nabla\langle E\rangle_i E$ then $M, \sigma \vDash \langle E\rangle_i E$. It is straightforward to prove by induction on n that $M, \sigma \vDash T_{L,P} \uparrow n$. Hence $M, \sigma \vDash I_{L,P}$. Therefore $P \vDash_L I_{L,P}$. •

Expected Lemma 3.3.5 *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.*

Proof. Let M be the standard L-model of I and σ the standard $\Diamond$-realization function on M. It is sufficient to prove that for any ground L-instance $\boxdot(A \leftarrow B_1, \ldots, B_n)$ of some program

clause of P, for any $w \in W$ being an L-instance of $\boxdot$, $M, w \vDash (A \leftarrow B_1, \ldots, B_n)$. Suppose that $M, w \vDash B_i$ for all $1 \leq i \leq n$. We show that $M, w \vDash A$.

Let $\triangle' = w$. We first show that for any ground simple atom B of the form E, $\Box_i E$, or $\Diamond_i E$, if $M, w \vDash B$ then $\triangle' B$ is an L-instance of some atom from $Sat_L(I)$. Suppose that $M, w \vDash B$. If B is of the form E, then by Lemma 3.2.1, some $\Box$-lifting form of $\triangle' B$ belongs to $Ext_L(I)$, and hence $\triangle' B$ is an L-instance of some atom from $Sat_L(I)$. If B is of the form $\Box_i E$ then, by the construction of M, it follows that $\Box_i E \in Ext_L(I)$, and hence $\{\Box_i E, \Box_m \Box_i E\} \subseteq Sat_L(I)$, which implies that $\triangle' B$ is an L-instance of some atom from $Sat_L(I)$. Now consider the case when B is of the form $\Diamond_i E$. Since $M, w \vDash \Diamond_i E$, either $\Box_j E \in Ext_L(I)$ or $\langle F \rangle_j E \in Ext_L(I)$ for some F and $j \leq i$. Hence, either $\{\Box_j E, \Box_m \Box_j E\} \subseteq Sat_L(I)$ or $\{\langle F \rangle_j E, \Box_m \Diamond_j E\} \subseteq Sat_L(I)$ for some F and $j \leq i$. Therefore $\triangle' B$ is an L-instance of some atom from $Sat_L(I)$.

Since $M, w \vDash B_i$ for $1 \leq i \leq n$, it follows that $\triangle' B_i$ is an L-instance of some atom from $Sat_L(I)$. Consequently, $\triangle' A$ is an L-instance of some atom α from $T_{0L,P}(Sat_L(I))$. Suppose that α is in L-normal form. We have $\alpha \in T_{L,P}(I) \subseteq I$. By Lemma 3.2.2, we have that $M, \sigma \vDash \alpha$, and hence $M, w \vDash A$. Now suppose that α is not in L-normal form, i.e. the length of the modality of α is greater than 1. Thus A is of the form $\Box_i E$ or $\Diamond_i E$. Let A' be the forward labeled form of A. We have $A' \in T_{L,P}(I)$. By Lemma 3.2.2, it follows that $M, \sigma \vDash A'$. Hence $M, w \vDash A$. •

Expected Lemma 3.5.1 *Let $\triangle$ and $\triangle'$ be ground modalities in L-normal labeled form. Let B be an atom of the form E, $\Diamond_i E$, or $\Box_i E$, and B' an atom of the form E, $\Diamond_j E$, $\langle X \rangle_j E$, or $\Box_j E$, where X is a fresh atom variable. Suppose that $\triangle$ is an L-instance of $\triangle'$ and B is an L-instance of B'. Then $\triangle' B'$ is derivable from $\triangle B$ using $rSat_L$.*

Proof. Because $\triangle$ and $\triangle'$ are modalities in L-normal labeled form and $\triangle$ is an L-instance of $\triangle'$, the atom $\triangle' B$ is derivable from $\triangle B$ using the $rSat_L$ rule "$\triangle \nabla_i \alpha \leftarrow \triangle \Box_j \alpha$ if $i \leq j$". Next, since B is an L-instance of B', $\triangle' B'$ is derivable from $\triangle' B$ using the first three rules specifying $rSat_L$. •

Expected Lemma 3.5.2 *Suppose that β is an atom in almost L-normal labeled form and $\alpha \in Sat_L(\{\beta\})$ or $\alpha \in NF_L(\{\beta\})$. Then there exists an atom β' and a substitution θ s.t. $\beta = \beta'\theta$, the domain of θ consists of fresh atom variables, and β' is derivable from α using $rSat_L$ and rNF_L.*

Proof. We give here a proof only for one representative case, when α is derived from β using the NF_L rule $\nabla \nabla' E \rightarrow \nabla' E$, where ∇' is of the form $\Box_i$ or $\langle E \rangle_i$. Suppose that $\alpha = \nabla' E$ and $\beta = \nabla \nabla' E$. If ∇ is of the form $\Box_j$, then by applying the rNF_L rule $\nabla' E \leftarrow \langle X \rangle_j \nabla' E$ and the $rSat_L$ (7) rule instance $\langle X \rangle_j \nabla' E \leftarrow \Box_j \nabla' E$ to α, we obtain $\beta' = \Box_j \nabla' E = \beta$. If ∇ is of the form $\langle F \rangle_j$ (resp. $\langle Y \rangle_j$), then by applying the rNF_L rule $\nabla' E \leftarrow \langle X \rangle_j \nabla' E$ to α, we obtain $\beta' = \langle X \rangle_j \nabla' E$ and have that $\beta = \beta'\theta$, where $\theta = \{X/F\}$ (resp. $\theta = \{X/Y\}$). •

Expected Lemma 3.5.3 *Let $\beta = rSat_L(\alpha)$, M be an L-model, σ a $\Diamond$-realization function on M, and θ a substitution. Suppose that $M, \sigma \vDash \forall_c(\beta'\theta)$ for some $\Box$-lifting form β' of β. Then $M, \sigma \vDash \forall_c(\alpha'\theta)$ for some $\Box$-lifting form α' of α.*

Proof. Consider the case when the rule used to derive β from α is $\nabla \nabla' E \leftarrow \nabla' E$, where ∇' is $\Box_i$ or $\Diamond_i$. Let $\alpha = \nabla_j \nabla' E$ and $\beta = \nabla' E$. Then we can choose $\alpha' = \Box_j \nabla' E$. It is easily seen that $M, \sigma \vDash \forall_c(\alpha'\theta)$, since $M, \sigma \vDash \forall_c(\beta'\theta)$. Now consider the case when the rule used to derive

β from α is $\triangle\Diamond_i E \leftarrow \triangle\Diamond_j E$ with $i > j$. Let $\alpha = \triangle\Diamond_i E$, $\beta = \triangle\Diamond_j E$, and $\beta' = \triangle'\nabla_j E$. Then we can choose $\alpha' = \triangle'\Diamond_i E$. Since $M, \sigma \vDash \forall_c(\beta'\theta)$, we have $M, \sigma \vDash \forall_c(\alpha'\theta)$. The two remaining cases are similar to the last case. •

Expected Lemma 3.5.4 *Let $\beta =_\delta rNF_L(\alpha)$, M be an L-model, σ a maximal $\Diamond$-realization function on M, and θ a substitution. Suppose that $M, \sigma \vDash \forall_c(\beta'\theta)$ for some $\Box$-lifting form β' of β. Then $M, \sigma \vDash \forall_c(\alpha'\delta\theta)$ for some $\Box$-lifting form α' of α.*

Proof. There is only one rNF_L rule. Let $\alpha\delta = \nabla E$ and $\beta = \langle X\rangle_j \nabla E$, where ∇ is $\Box_i$ or $\langle E\rangle_i$. If $\nabla = \Box_i$, then let $\nabla' = \Box_i$, else let $\nabla' = \Diamond_i$. Since $M, \sigma \vDash \forall_c(\beta'\theta)$, we have $M \vDash \nabla' E\theta$. Since σ is a maximal $\Diamond$-realization function on M, it follows that $M, \sigma \vDash \forall_c(\nabla E\theta)$. Hence we can choose $\alpha' = \alpha$. •

4.3.3 A Schema for Semantics of $KDI45$-MProlog

In this subsection, L denotes the logic $KDI45$. We first show that $\nabla_i \nabla'_j \alpha \equiv \nabla'_j \alpha$ is L-valid for $i \leq j$, where ∇_i and ∇'_j are unlabeled modal operators. Suppose that $i \leq j$. There are the following cases:

- Case $\nabla_i = \Box_i$ and $\nabla'_j = \Box_j$: $\Box_i\Box_j\alpha \rightarrow \Box_j\alpha$ follows from $\Box_i\Box_j\alpha \rightarrow \Diamond_i\Box_j\alpha$, $\Diamond_i\Box_j\alpha \rightarrow \Diamond_j\Box_j\alpha$, and $\Diamond_j\Box_j\alpha \rightarrow \Box_j\alpha$. For the conversion, $\Box_j\alpha \rightarrow \Box_i\Box_j\alpha$ follows from $\Box_j\alpha \rightarrow \Box_j\Box_j\alpha$ and $\Box_j\Box_j\alpha \rightarrow \Box_i\Box_j\alpha$.
- Case $\nabla_i = \Box_i$ and $\nabla'_j = \Diamond_j$: $\Box_i\Diamond_j\alpha \rightarrow \Diamond_j\alpha$ follows from $\Box_i\Diamond_j\alpha \rightarrow \Diamond_i\Diamond_j\alpha$, $\Diamond_i\Diamond_j\alpha \rightarrow \Diamond_j\Diamond_j\alpha$, and $\Diamond_j\Diamond_j\alpha \rightarrow \Diamond_j\alpha$. For the conversion, $\Diamond_j\alpha \rightarrow \Box_i\Diamond_j\alpha$ follows from $\Diamond_j\alpha \rightarrow \Box_j\Diamond_j\alpha$ and $\Box_j\Diamond_j\alpha \rightarrow \Box_i\Diamond_j\alpha$.
- Case $\nabla_i = \Diamond_i$ and ($\nabla'_j = \Box_j$ or $\nabla'_j = \Diamond_j$) : These cases are dual to the above cases.

The equivalence suggests that a modality $\nabla^{(1)}_{i_1} \ldots \nabla^{(k)}_{i_k}$ is in L-normal form if $i_1 > \ldots > i_k$.

In Table 4.6, we give a schema for semantics of $KDI45$-MProlog. The rule (1) follows from the axiom (I); the rule (2) from the axioms (4) and (I); the rule (3) from the axioms (I), (D), and (5) (because $\triangle\Box_i\Box_j\alpha \rightarrow \triangle\Box_j\Box_j\alpha$, $\triangle\Box_j\Box_j\alpha \rightarrow \triangle\Diamond_j\Box_j\alpha$, and $\triangle\Diamond_j\Box_j\alpha \rightarrow \triangle\Box_j\alpha$); the rule (4) follows from the axiom (4); the rule (5) is based on the axiom (5); the rules (6) and (7) are based on the axiom (4) (as $\triangle\Diamond_k\Diamond_k E \rightarrow \triangle\Diamond_k E$); the rules (8) and (9) are based on the mentioned observation on L-normal form of modalities; the rule (11) is based on the axioms (D) and (I); the rule (12) is based on the axiom (I); the rule (13) is dual to the rules (2) and (4); the rule (14) is dual to the rule (5); the rule (15) is dual to the rule (3); and finally, the rule (16) is based on the axiom (4) and dual to the rules (6) and (7).

Note that if ∇_i is a maximally general L-instance of labeled modal operators ∇'_j and ∇''_k then $i = min(j, k)$. It follows that if $\triangle$ is a maximally general L-instance of $\triangle^{(1)}, \ldots, \triangle^{(k)}$, which are modalities in L-normal labeled form, then $\triangle$ is also in L-normal labeled form.

Also note that a universal modality $\boxdot$ in L-normal form is an L-context instance of $\Box_i$ iff $\boxdot$ is not empty and every modal operator $\Box_j$ of $\boxdot$ satisfies $j \leq i$.

Theorem 4.3.3 *The schema given in Table 4.6 for semantics of $KDI45$-MProlog is correct.*

To prove this theorem we have to prove Expected Theorem 3.2.3 and Expected Lemmas 3.2.2, 3.3.1, 3.3.2, 3.3.4, 3.3.5, 3.5.1 – 3.5.4.

Expected Lemmas 3.3.1 (page 35) and 3.5.1 – 3.5.4 (page 40) can easily be verified.

$L = KDI45$, L-MProlog

$\preceq_L$ is defined by Definition 3.3.1.

$\nabla_{i_1}^{(1)} \ldots \nabla_{i_k}^{(k)}$ is in L-normal form if $i_1 > \ldots > i_k$.

The following rules are accompanied with the condition that the atoms in both sides are in L-normal labeled form for rules specifying Ext_L and in almost L-normal labeled form for the other rules. (*)

Rules specifying

Ext_L
$\triangle\Box_i\alpha \rightarrow \triangle\Box_j\alpha$ if $i > j$ (1)
$\triangle\Box_i\alpha \rightarrow \triangle\Box_i\Box_j\alpha$ if $i > j$ (2)
$\triangle\Box_i\Box_j\alpha \rightarrow \triangle\Box_j\alpha$ if $i > j$ (3)

Sat_L the rules for Ext_L with the modification stated in (*), plus
$\triangle\Box_i E \rightarrow \triangle\Box_i\Box_i E$ (4)
$\triangle\nabla E \rightarrow \triangle\Box_i\Diamond_i E$ if $\Diamond_i \preceq_L \nabla$ (5)
$\triangle\Box_i\nabla_j E \rightarrow \triangle\Diamond_j E$ if $i > j$ (6)
$\triangle\langle F\rangle_i\nabla_j E \rightarrow \triangle\Diamond_i E$ if $i > j$ (7)

NF_L $\triangle\nabla_i\nabla'_j E \rightarrow \triangle\nabla'_j E$ if ∇'_j is of the form $\Box_j$ or $\langle E\rangle_j$ and $i \leq j$ (8)

rNF_L $\triangle\nabla_j E \leftarrow \triangle\langle X\rangle_i\nabla_j E$ if ∇_j is of the form $\Box_j$ or $\langle E\rangle_j$, X is a fresh atom variable, and $i \leq j$ (9)

$rSat_L$
$\triangle\Diamond_i E \leftarrow \triangle\langle X\rangle_i E$ for X being a fresh atom variable (10)
$\triangle\nabla_i\alpha \leftarrow \triangle\Box_j\alpha$ if $i \leq j$ (11)
$\triangle\Diamond_i E \leftarrow \triangle\Diamond_j E$ if $i > j$ (12)
$\triangle\Box_i\Box_j\alpha \leftarrow \triangle\Box_i\alpha$ if $i \geq j$ (13)
$\triangle\Box_i\Diamond_i E \leftarrow \triangle\Diamond_i E$ (14)
$\triangle\Box_i\alpha \leftarrow \triangle\Box_j\Box_i\alpha$ if $i < j$ (15)
$\triangle\Diamond_i E \leftarrow \triangle\langle X\rangle_i\Diamond_i E$ for X being a fresh atom variable (16)

Table 4.6: A schema for semantics of $KDI45$-MProlog

Expected Lemma 3.2.2 *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 $M, \sigma \models I$.*

Proof. By the definition, M is an L-model. Let $(R'_i)_{1 \leq i \leq m}$ be the skeleton of M. We prove by induction on the length of α that for any $w \in W$, if $\alpha \in H(w)$, then $M, \sigma, w \models \alpha$. The cases when α is a classical atom or $\alpha = \langle E \rangle_i \beta$ are trivial. Suppose that $\alpha = \Box_i \beta$. Let u be a world such that $R_i(w, u)$ holds. We show that $\beta \in H(u)$. Since $R_i(w, u)$ holds, there exist worlds $w_0, \ldots, w_k$ ($k \geq 0$) and $u_0, \ldots, u_h$ ($h \geq 1$) such that $w_0 = u_0$, $w_k = w$, $u_h = u$, $R'_{i_t}(w_{t-1}, w_t)$ for $1 \leq t \leq k$ with $i \geq i_1 > \ldots > i_k$, and $R'_{j_s}(u_{s-1}, u_s)$ for $1 \leq s \leq h$ with $i \geq j_1 > \ldots > j_h$. On the other hand, because that all atoms of $H(w_0)$ are in L-normal form and $\Box_i \beta \in H(w)$ (i.e. $\Box_i \beta \in H(w_k)$), k must be 0. Hence $\Box_i \beta \in H(u_0)$. By Lemma 3.2.1, there exists a $\Box$-lifting form $\triangle$ of u_0 such that $\triangle \Box_i \beta \in Ext_L(I)$. By the definition of Ext_L, it follows that $\triangle \Box_{j_1} \ldots \Box_{j_h} \beta \in Ext_L(I)$. Hence, by Lemma 3.2.1, $\beta \in H(u)$. •

Expected Theorem 3.2.3 *The standard L-model of an L-normal model generator I is a least L-model of I.*

Proof. Let $M = \langle W, \tau, R_1, \ldots, R_m, H \rangle$ be the standard L-model graph of I, σ the standard $\Diamond$-realization function and $(R'_i)_{1 \leq i \leq m}$ the skeleton of the standard L-model of I. By Lemma 3.2.2, M is an L-model of I. Let $M' = \langle W, \tau, R_1, \ldots, R_m, H' \rangle$ be the least extension of M such that if $\Box_i \alpha \in H'(w)$, $R'_j(w, u)$ and $j \leq i$ then $\Box_i \alpha \in H'(u)$. Let $N = \langle D, W_2, \tau_2, S_1, \ldots, S_m, \pi \rangle$ be an arbitrary L-model of I and σ_2 a $\Diamond$-realization function on N such that $N, \sigma_2 \models I \cup \{ \boxdot \langle \top \rangle_i \mid \boxdot$ is a universal modality and $1 \leq i \leq m \}$.

Let $r \subseteq W \times W_2$ be the least relation such that, for all w, w_2, u_2, E, i :

- $r(\tau, \tau_2)$;
- if $r(w, w_2)$ and $R'_i(w, w\langle E \rangle_i)$ hold and $\sigma_2(w_2, \langle E \rangle_i)$ is defined then $r(w\langle E \rangle_i, \sigma_2(w_2, \langle E \rangle_i))$;
- if $r(w, w_2)$ and $S_i(w_2, u_2)$ hold and $w = \langle E_1 \rangle_{i_1} \ldots \langle E_k \rangle_{i_k}$ with $k \geq 0$, then $r(u, u_2)$ for $u = \langle E_1 \rangle_{i_1} \ldots \langle E_j \rangle_{i_j} \langle \top \rangle_i$ with $0 \leq j \leq k$ such that $i_j > i$ (if $j > 0$) and $i_{j+1} \leq i$ (if $j < k$).

The last condition in the above definition of r implies that: if $r(w, w_2)$ and $S_i(w_2, u_2)$ hold then there exists u such that $r(u, u_2)$ and $R_i(w, u)$.

We prove that $M' \leq_r N$. We first show that if $r(u, u_2)$ and $\alpha \in H'(u)$ then $N, \sigma_2, u_2 \models \alpha$. Suppose that $r(u, u_2)$ holds and $\alpha \in H'(u)$. We prove the assertion by induction on the number of the step in the construction of r at which $r(u, u_2)$ is created. The case $u = \tau$ is trivial. There are two remaining cases:

- $u = w\langle E \rangle_i$, $u_2 = \sigma_2(w_2, \langle E \rangle_i)$, $r(w, w_2)$, and $R'_i(w, w\langle E \rangle_i)$; or
- $u = \langle E_1 \rangle_{i_1} \ldots \langle E_j \rangle_{i_j} \langle \top \rangle_i$, $w = \langle E_1 \rangle_{i_1} \ldots \langle E_k \rangle_{i_k}$, $0 \leq j \leq k$, $i_j > i$ (if $j > 0$), $i_{j+1} \leq i$ (if $j < k$), $r(w, w_2)$, and $S_i(w_2, u_2)$.

Consider the first case. Since $\alpha \in H'(u)$, there are three subcases: i) $\Box_i \alpha \in H'(w)$, ii) $\langle E \rangle_i \alpha \in H'(w)$, iii) α is of the form $\Box_j \beta$, $j \geq i$, and $\Box_j \beta \in H'(w)$. For the first two subcases, by the inductive assumption, either $N, \sigma_2, w_2 \models \Box_i \alpha$ or $N, \sigma_2, w_2 \models \langle E \rangle_i \alpha$, and hence $N, \sigma_2, \sigma_2(w_2, \langle E \rangle_i) \models \alpha$, which means that $N, \sigma_2, u_2 \models \alpha$. Consider the last subcase. By the inductive assumption, $N, \sigma_2, w_2 \models \Box_j \beta$, and hence $N, \sigma_2, w_2 \models \Box_i \Box_j \beta$ (by the axioms (4) and (I)). Therefore $N, \sigma_2, \sigma_2(w_2, \langle E \rangle_i) \models \Box_j \beta$, which means that $N, \sigma_2, u_2 \models \alpha$.

Consider the second case. Let v be the world such that $u = v\langle\top\rangle_i$. Since $\alpha \in H'(u)$, either $\Box_i\alpha \in H'(v)$ or α is of the form $\Box_j\beta$, $j \geq i$, and $\Box_j\beta \in H'(v)$. Suppose that $\Box_i\alpha \in H'(v)$. It follows that $\Box_i\alpha \in H'(w)$. By the inductive assumption, $N, \sigma_2, w_2 \vDash \Box_i\alpha$, and hence $N, \sigma_2, u_2 \vDash \alpha$ since $S_i(w_2, u_2)$. Now suppose that α is of the form $\Box_j\beta$, $j \geq i$, and $\Box_j\beta \in H'(v)$. We have $\Box_j\beta \in H'(w)$. By the inductive assumption, $N, \sigma_2, w_2 \vDash \Box_j\beta$, and hence $N, \sigma_2, u_2 \vDash \Box_j\beta$ since $S_i(w_2, u_2)$ and $j \geq i$. This means that $N, \sigma_2, u_2 \vDash \alpha$.

We now show that if $r(w, w_2)$ and $R'_i(w, w\langle E\rangle_i)$ hold then $\sigma_2(w_2, \langle E\rangle_i)$ is defined. The case $E = \top$ is trivial. Suppose that $r(w, w_2)$ and $R'_i(w, w\langle E\rangle_i)$ hold and $E \neq \top$. Thus, there exists $\langle E\rangle_i\alpha \in H'(w)$ for some α. By the first assertion, $N, \sigma_2, w_2 \vDash \langle E\rangle_i\alpha$. Hence $\sigma_2(w_2, \langle E\rangle_i)$ is defined. Therefore, the second condition in the above definition of r can be simplified to "if $r(w, w_2)$ and $R'_i(w, w\langle E\rangle_i)$ hold then $r(w\langle E\rangle_i, \sigma_2(w_2, \langle E\rangle_i))$".

Next, we show that if $r(u, u_2)$ holds and $u = w\langle E\rangle_i$ then there exists w_2 such that $r(w, w_2)$ and $S_i(w_2, u_2)$. We prove this by induction on the construction of r. Suppose that $r(u, u_2)$ and $u = w\langle E\rangle_i$ hold. If $r(u, u_2)$ is created by the second condition of the definition of r, then the assertion clearly holds. Suppose that $r(u, u_2)$ is created by the third condition of the definition of r, i.e. $r(u, u_2)$ is created from $r(v, v_2)$ with the property that $u = w\langle\top\rangle_i$, $S_i(v_2, u_2)$, $v = w\langle E_1\rangle_{i_1} \ldots \langle E_k\rangle_{i_k}$, $k \geq 0$, $i \geq i_1 > i_2 > \ldots > i_k$. By the inductive assumption for $r(v, v_2)$, there exist w_2 such that $r(w, w_2)$ and a path from w_2 to v_2 consisting of edges connected via $S_{i_1}, \ldots, S_{i_k}$. Thus $S_i(w_2, u_2)$ holds.

To prove that $M' \leq_r N$ there remains to show that if $r(w, w')$ and $R_i(w, u)$ hold, then there exists $u' \in W_2$ such that $r(u, u')$ and $S_i(w', u')$. Suppose that $r(w, w')$ and $R_i(w, u)$ hold. Since $R_i(w, u)$ holds, there exist worlds $w_0, \ldots, w_k$ ($k \geq 0$) and $u_0, \ldots, u_h$ ($h \geq 1$) such that $w_0 = u_0$, $w_k = w$, $u_h = u$, $R'_{i_t}(w_{t-1}, w_t)$ for $1 \leq t \leq k$ with $i \geq i_1 > \ldots > i_k$, and $R'_{j_s}(u_{s-1}, u_s)$ for $1 \leq s \leq h$ with $i \geq j_1 > \ldots > j_h$. Since $r(w, w')$ holds, by the assertion proved in the previous paragraph, there exists w'_0 such that $r(w_0, w'_0)$ and $S_i(w'_0, w')$. Hence, by the construction of r, there exists u' such that $r(u, u')$ and $S_i(w'_0, u')$. Thus we have that $r(u, u')$ and $S_i(w', u')$.

Therefore $M' \leq_r N$ and the standard L-model graph of I is a least L-model of I. •

Expected Lemma 3.3.2 *Let I be an L-normal model generator, M the standard L-model of I, and α a ground L-MProlog goal atom. Suppose that $M \vDash \alpha$. Then α is an L-instance of some atom of $Sat_L(I)$.*

Proof. If α is of the form E or $\Box_i E$, then $\alpha \in Ext_L(I)$ (since $M \vDash \alpha$), and hence $\alpha \in Sat_L(I)$. Suppose that $\alpha = \Diamond_i E$. Let $\langle W, \tau, R_1, \ldots, R_m, H\rangle$ be the standard L-model graph of I. Since $M \vDash \alpha$, there exists a world $u = \langle E_1\rangle_{i_1} \ldots \langle E_k\rangle_{i_k}$ (with $k \geq 1$) of M such that $i \geq i_1 > \ldots > i_k$ and $E \in H(u)$. By Lemma 3.2.1, some $\Box$-lifting form of $\langle E_1\rangle_{i_1} \ldots \langle E_k\rangle_{i_k} E$ belongs to $Ext_L(I)$, and hence also to $Sat_L(I)$. By the definition of Sat_L, it follows that there exists some $\Diamond_j E \in Sat_L(I)$ with $j \leq i$. Hence α is an L-instance of some atom from $Sat_L(I)$. •

Expected Lemma 3.3.4 *If P is an L-MProlog program then $P \vDash_L I_{L,P}$.*

Proof. Let $M = \langle D, W, \tau, R_1, \ldots, R_m, \pi\rangle$ be an arbitrary L-model of P and σ a maximal $\Diamond$-realization function on M. Note that if $M, \sigma \vDash \triangle\langle E\rangle_i E$ then $M, \sigma \vDash NF_L(\{\triangle\langle E\rangle_i E\})$. It is straightforward to prove by induction on n that $M, \sigma \vDash T_{L,P}\uparrow n$. Therefore $M, \sigma \vDash I_{L,P}$ and $P \vDash_L I_{L,P}$. •

Expected Lemma 3.3.5 *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.*

Proof. Let M be the standard L-model of I and σ the standard $\Diamond$-realization function on M. By the definition of L-instances of program clauses and the construction of M, it is sufficient to prove that for any ground L-instance $\boxdot(A \leftarrow B_1, \ldots, B_n)$ of some program clause of P and for any $w \in W$ being an L-instance of $\boxdot$, $M, w \vDash (A \leftarrow B_1, \ldots, B_n)$. Suppose that $M, w \vDash B_i$ for all $1 \leq i \leq n$. We show that $M, w \vDash A$.

Let $\triangle = w$. We first show that for any ground simple atom B of the form E, $\Box_i E$, or $\Diamond_i E$, if $M, w \vDash B$ then $\triangle B$ is an L-instance of some atom from $Sat_L(I)$. Suppose that $M, w \vDash B$. If $B = E$, then by Lemma 3.2.1, some $\Box$-lifting form of $\triangle B$ belongs to $Ext_L(I)$, and hence $\triangle B$ is an L-instance of some atom from $Sat_L(I)$. Now suppose that $B = \Box_i E$. Let $w = \langle E_1 \rangle_{i_1} \ldots \langle E_k \rangle_{i_k}$ and $0 \leq j \leq k$ be the index such that $i_j > i$ (if $j > 0$) and $i \geq i_{j+1}$ (if $j < k$). Let $u = \langle E_1 \rangle_{i_1} \ldots \langle E_j \rangle_{i_j} \langle \top \rangle_i$. We have that $R_i(w, u)$, and hence $M, u \vDash E$. By Lemma 3.2.1, it follows that some $\Box$-lifting form of $\langle E_1 \rangle_{i_1} \ldots \langle E_j \rangle_{i_j} \Box_i E$ belongs to $Ext_L(I)$. Hence, by the definition of Sat_L, $\triangle B$ is an L-instance of some atom from $Sat_L(I)$. Next, suppose that $B = \Diamond_i E$. Let u be a world such that $R_i(w, u)$ and $M, u \vDash E$. Since $R_i(w, u)$ holds, there exists $\triangle'$, $\triangle''$, $\triangle'''$ such that $w = \triangle' \triangle''$, $u = \triangle' \triangle'''$, $\triangle'$ and $\triangle''$ can be empty while $\triangle'''$ cannot, $\triangle''$ and $\triangle'''$ contain only modal operators with index (i.e. degree) less than or equal to i, while $\triangle'$ contains only modal operators with index greater than i. Since $M, u \vDash E$, by Lemma 3.2.1, some $\Box$-lifting form of $\triangle' \triangle''' E$ belongs to $Ext_L(I)$. Hence, by the definition of Sat_L, there exists $j \leq i$ such that some $\Box$-lifting form of $\triangle' \Diamond_j E$ belongs to $Sat_L(I)$. If $\triangle''$ is not empty then, by the definition of Sat_L, it follows that some $\Box$-lifting form of $\triangle' \triangle'' \Diamond_i E$ belongs to $Sat_L(I)$. Hence $\triangle B$ is an L-instance of some atom from $Sat_L(I)$.

Since $M, w \vDash B_i$ for $1 \leq i \leq n$, it follows that $\triangle B_i$ is an L-instance of some atom from $Sat_L(I)$. Consequently, $\triangle A$ is an L-instance of some atom α from $T_{0L,P}(Sat_L(I))$. Let α' be the L-normal form of α, i.e. $NF_L(\{\alpha\}) = \{\alpha'\}$. We have $\alpha' \in T_{L,P}(I) \subseteq I$. By Lemma 3.2.2, $M, \sigma \vDash \alpha'$. From this we can conclude that $M, w \vDash A$. •

4.4 Programming in MProlog for Multi-agent Systems

To program for multi-agent systems we can use the logics $KD4_s5_s$, $KD45_{(m)}$, and $KD4I_g5_a$. In these logics, $\Box_i \varphi$ stands for "agent i believes that φ is true", while $\Diamond_i \varphi$ stands for "φ is considered possible by agent i". The logic $KD4_s5_s$ can be used for distributed systems of belief, in which agents have full access to belief bases of each other. The logics $KD45_{(m)}$ and $KD4I_g5_a$ are intended for reasoning about epistemic states of agents. In $KD4I_g5_a$, some modal indices stand for groups of agents, and using them we can reason about common belief. In this section, we present schemata for semantics of MProlog in the three mentioned logics and prove their correctness.

4.4.1 A Schema for Semantics of $KD4_s5_s$-MProlog

In this subsection, L denotes the logic $KDI4_s5$. It can be checked that a connected frame $\langle W, \tau, R_1, \ldots, R_m \rangle$ is a $KD4_s5_s$-frame iff there are nonempty subsets of worlds $W_1, \ldots, W_m$ such that $W = \{\tau\} \cup W_1 \cup \ldots \cup W_m$ and $R_i = W \times W_i$, for $1 \leq i \leq m$. Note that this property is similar to the property of $KDI4_s5$-frames. The difference is that the logic $KD4_s5_s$ does not contain the axiom (I) and in this logic we do not have the condition that $W_i \subseteq W_j$ for $i < j$.

In Table 4.7, we present a schema for semantics of $KD4_s5_s$-MProlog. L-normal form of modalities and the rules (1)–(4) and (7) in that schema are justified by the L-tautology $\nabla\varphi \equiv \nabla'\nabla\varphi$ with ∇ and ∇' being unlabeled modal operators, while the rule (6) is based

$L = KD4_s5_s$, L-MProlog		
$\preceq_L$ is defined by Definition 3.3.1.		
A modality is in L-normal form if its length ≤ 1.		
Rules specifying		
Ext_L	no rules	
Sat_L	$\Box_i E \to \Box_j \Box_i E$	(1)
	$\langle F \rangle_i E \to \Box_j \Diamond_i E$	(2)
NF_L	$\nabla\nabla' E \to \nabla' E$ if ∇' is of the form $\Box_i$ or $\langle E \rangle_i$	(3)
rNF_L	$\nabla E \leftarrow \langle X \rangle_j \nabla E$ if ∇ is of the form $\Box_i$ or $\langle E \rangle_i$ and X is a fresh atom variable	(4)
$rSat_L$	$\triangle\Diamond_i E \leftarrow \triangle\langle X \rangle_i E$ for X being a fresh atom variable	(5)
	$\triangle\nabla_i \alpha \leftarrow \triangle\Box_i \alpha$	(6)
	$\nabla\nabla' E \leftarrow \nabla' E$ if ∇' is of the form $\Box_i$ or $\Diamond_i$	(7)

Table 4.7: A schema for semantics of $KD4_s5_s$-MProlog

on the axiom (D). This schema is similar to the schema for semantics of $KDI4_s5$-MProlog, except that it does not contain rules involving with the axiom (I). Note that $\boxdot$ is an L-context instance of $\boxdot'$ iff $\boxdot = \boxdot'$ (here, $|\boxdot| = |\boxdot'| = 1$). Analogously as for $KDI4_s5$, we can prove the following theorem.

Theorem 4.4.1 *The schema given in Table 4.7 for semantics of $KD4_s5_s$-MProlog is correct.*

Example 4.4.1 Reconsider the MProlog program P_{ddb} given in Example 2.9.2. To increase readability, we recall some clauses of P_{ddb}:

$\varphi_1 = \Box_1 likes(Jan, cola) \leftarrow$
$\varphi_5 = \Box_2 likes(Jan, pepsi) \leftarrow$
$\varphi_8 = \Box_2(likes(x, cola) \leftarrow likes(x, pepsi))$
$\varphi_{10} = \Box_3 likes(Jan, cola) \leftarrow$
$\varphi_{13} = \Box_3(very\_much\_likes(x, y) \leftarrow likes(x, y), \Box_1 likes(x, y), \Box_2 likes(x, y))$
$\varphi_{14} = very\_much\_likes(x, y) \leftarrow \Box_3 very\_much\_likes(x, y)$

Here is an SLD-refutation of $P \cup \{\leftarrow very\_much\_likes(x, y)\}$ in $KD4_s5_s$:

Goals	Input clauses/rules	MGUs
$\leftarrow very\_much\_likes(x, y)$		
$\leftarrow \Box_3 very\_much\_likes(x, y)$	φ_{14}	$\{x_1/x, y_1/y\}$
$\leftarrow \Box_3 likes(x, y), \Box_3\Box_1 likes(x, y), \Box_3\Box_2 likes(x, y)$	φ_{13}	$\{x_2/x, y_2/y\}$

$\leftarrow \Box_3\Box_1 likes(Jan, cola), \Box_3\Box_2 likes(Jan, cola)$	φ_{10}	$\{x/Jan, y/cola\}$
$\leftarrow \Box_1 likes(Jan, cola), \Box_3\Box_2 likes(Jan, cola)$	(7)	ε
$\leftarrow \Box_3\Box_2 likes(Jan, cola)$	φ_1	ε
$\leftarrow \Box_2 likes(Jan, cola)$	(7)	ε
$\leftarrow \Box_2 likes(Jan, pepsi)$	φ_8	$\{x_7/Jan\}$
$\diamond$	φ_5	ε

The schema given in Table 4.7 is formulated so that it can use the proofs given in Section 3. However, similarly as for the case of $KDI4_s5$, the rules (5) and (6) of Table 4.7 can be simplified in the way that the occurrences of $\triangle$ in those rules are deleted and α in the rule (6) is replaced by E. Furthermore, when resolving a goal with an input clause, if we relax the condition that "the mgu θ unifies the selected head atom A' with the forward labeled form A'' of the head of the input clause" by requiring only that θ is a most general substitution such that $A'\theta$ and $A''\theta$ have the same classical atom and $A'\theta$ is an L-instance of $A''\theta$, then the rule (6) can be deleted. It can be shown that every SLD-refutation in the original calculus can be simulated in the new calculus by another one with the same computed answer. This means that the new SLD-resolution calculus is also sound and complete.

An agent should keep clauses that define its epistemic states. This means that agent i should keep clauses of the form $\nabla_i E \leftarrow B_1, \ldots, B_n$ or $\Box_i(A \leftarrow B_1, \ldots, B_n)$. Furthermore, program clauses of the form $\Box_i(\Box_j E \leftarrow B_1, \ldots, B_n)$ with $i \neq j$ have little sense in distributed systems of belief. It can be shown that program clauses of that form can be disallowed without reducing expressiveness of $KD4_s5_s$-MProlog. If we adopt this restriction then the rule (4) in Table 4.7 can be modified so that the involved modal operators have the same modal index (i.e. agent index). Program clauses of the form $E \leftarrow B_1, \ldots, B_n$ can be kept by a special agent, which communicates with users. Whenever an agent meets a goal atom of the form $\nabla_i E$ it will require agent i to solve the goal $\leftarrow \nabla_i E$, and whenever an agent meets a goal atom of the form E (without modal context) it will require the special agent to solve the goal $\leftarrow E$.

4.4.2 A Schema for Semantics of $KD45_{(m)}$-MProlog

In this subsection, L denotes the logic $KD45_{(m)}$. Recall that this logic is intended for reasoning about epistemic states of a number of agents. This logic is weak at the aspect that it has no interaction axioms between agents. On the other hand, this restriction together with L-normal form of modalities prevents expanding modal contexts of goal atoms.

In Table 4.8, we present a schema for semantics of $KD45_{(m)}$-MProlog. L-normal form of modalities and the rules (1)–(4) and (7) in that schema are justified by the L-tautology $\nabla_i\varphi \equiv \nabla'_i\nabla_i\varphi$, where ∇_i and ∇'_i are unlabeled modal operators. The rule (6) is based on the axiom (D). Note that $\boxdot$ is an L-context instance of $\boxdot'$ iff $\boxdot = \boxdot'$.

The schema given in Table 4.8 is formulated so that it can use the proofs given in Section 3. However, the rule (6) can be deleted without violating soundness and completeness of SLD-resolution if: a) the condition of the rule (4) that ∇ is of the form $\Box_i$ or $\langle E\rangle_i$ is deleted, b) when resolving a goal with an input clause, we relax the condition that "the mgu θ unifies the selected head atom A' with the forward labeled form A'' of the head of the input clause" by requiring only that θ is a most general substitution such that $A'\theta$ and $A''\theta$ have the same classical atom and $A'\theta$ is an L-instance of $A''\theta$. It can be shown that every SLD-refutation in the original calculus can be simulated in the new calculus by another one with a more general computed answer, and vice versa. This means that the new SLD-resolution calculus is sound and complete, provided that so is the original calculus.

$L = KD45_{(m)}$, L-MProlog		
$\preceq_L$ is defined by Definition 3.3.1.		
$\nabla_{i_1}^{(1)} \dots \nabla_{i_k}^{(k)}$ is in L-normal form if $i_j \neq i_{j+1}$ for all $1 \leq j < k$.		
Both sides of each rule given below are in almost L-normal labeled form.		
Rules specifying		
Ext_L	no rules	
Sat_L	$\triangle\Box_i E \to \triangle\Box_i\Box_i E$	(1)
	$\triangle\langle F\rangle_i E \to \triangle\Box_i\Diamond_i E$	(2)
NF_L	$\triangle\nabla_i\nabla'_i E \to \triangle\nabla'_i E$ if ∇'_i is of the form $\Box_i$ or $\langle E\rangle_i$	(3)
rNF_L	$\triangle\nabla_i E \leftarrow \triangle\langle X\rangle_i\nabla_i E$ if ∇'_i is of the form $\Box_i$ or $\langle E\rangle_i$ and X is a fresh atom variable	(4)
$rSat_L$	$\triangle\Diamond_i E \leftarrow \triangle\langle X\rangle_i E$ for X being a fresh atom variable	(5)
	$\triangle\nabla_i\alpha \leftarrow \triangle\Box_i\alpha$	(6)
	$\triangle\nabla_i\nabla'_i E \leftarrow \triangle\nabla'_i E$ if ∇'_i is of the form $\Box_i$ or $\Diamond_i$	(7)

Table 4.8: A schema for semantics of $KD45_{(m)}$-MProlog

Theorem 4.4.2 *The schema given in Table 4.8 for semantics of $KD45_{(m)}$-MProlog is correct.*

To prove this theorem we have to prove Expected Theorem 3.2.3 and Expected Lemmas 3.2.2, 3.3.1, 3.3.2, 3.3.4, 3.3.5, 3.5.1 – 3.5.4.

Expected Lemmas 3.3.1 (page 35) and 3.5.1 – 3.5.4 (page 40) can easily be verified. The proof of Expected Lemma 3.3.4 is the same as for $KDI45$ (see page 67).

Expected Lemma 3.2.2 *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 $M, \sigma \vDash I$.*

Proof. By the definition, M is an L-model. Let $(R'_i)_{1\leq i\leq m}$ be the skeleton of M. We prove by induction on the length of α that for any $w \in W$, if $\alpha \in H(w)$ then $M, \sigma, w \vDash \alpha$. The cases when α is a classical atom or $\alpha = \langle E\rangle_i\beta$ are trivial. Consider the remaining case when $\alpha = \Box_i\beta$. Let u be a world such that $R_i(w, u)$ holds. Due to L-normal form of modalities, $\Box_i\beta \in H(w)$ and $R_i(w, u)$ imply that $R'_i(w, u)$ holds, and hence $\beta \in H(u)$. By the inductive assumption, $M, \sigma, u \vDash \beta$. Hence $M, \sigma, w \vDash \alpha$. •

Expected Theorem 3.2.3 *The standard L-model of an L-normal model generator I is a least L-model of I.*

Proof. Let $M = \langle W, \tau, R_1, \ldots, R_m, H\rangle$ be the standard L-model graph of I, σ the standard $\Diamond$-realization function and $(R'_i)_{1\leq i\leq m}$ the skeleton of the standard L-model of I. By Lemma 3.2.2, M is an L-model of I. Let $N = \langle D, W_2, \tau_2, S_1, \ldots, S_m, \pi\rangle$ be an arbitrary L-model of I and σ_2 a $\Diamond$-realization function on N such that $N, \sigma_2 \vDash I \cup \{\boxdot\langle\top\rangle_i \mid \boxdot$ is a universal modality and $1 \leq i \leq m\}$.

Let $r \subseteq W \times W_2$ be the least relation such that, for all w, w_2, u_2, E, i :

- $r(\tau, \tau_2)$;
- if $r(w, w_2)$ and $R'_i(w, w\langle E\rangle_i)$ hold and $\sigma_2(w_2, \langle E\rangle_i)$ is defined then $r(w\langle E\rangle_i, \sigma_2(w_2, \langle E\rangle_i))$;
- if $r(w, w_2)$ and $S_i(w_2, u_2)$ hold then $r(u, u_2)$ holds for u such that: if $w = \tau$ then $u = \langle\top\rangle_i$, if $w = v\langle E\rangle_j$ and $j \neq i$ then $u = w\langle\top\rangle_i$, and if $w = v\langle E\rangle_i$ then $u = v\langle\top\rangle_i$.

The last condition in the above definition of r implies that: if $r(w, w_2)$ and $S_i(w_2, u_2)$ hold then there exists u such that $r(u, u_2)$ and $R_i(w, u)$.

It is straightforward to prove by induction on the construction of r that, if $R'_i(x, y)$ and $r(y, y_2)$ then there exists x_2 such that $r(x, x_2)$ and $S_i(x_2, y_2)$. Denote this assertion by (*).

We prove that $M \leq_r N$. For this we first show that if $r(u, u_2)$ and $\alpha \in H(u)$ then $N, \sigma_2, u_2 \vDash \alpha$. We prove this by induction on the length of u. The case $u = \tau$ is trivial. Suppose that $r(u, u_2)$, $u \neq \tau$, and $\alpha \in H(u)$. There are the following cases to consider:

- Case $u = w\langle E\rangle_i$, $u_2 = \sigma_2(w_2, \langle E\rangle_i)$, and $r(w, w_2)$: Since $\alpha \in H(u)$, either $\Box_i\alpha \in H(w)$ or $\langle E\rangle_i\alpha \in H(w)$. Hence, by the inductive assumption, $N, \sigma_2, w_2 \vDash \Box_i\alpha$ or $N, \sigma_2, w_2 \vDash \langle E\rangle_i\alpha$. It follows that $N, \sigma_2, u_2 \vDash \alpha$.
- Case $u = \langle\top\rangle_i$ and $S_i(\tau_2, u_2)$: Since $\alpha \in H(u)$, it follows that $\Box_i\alpha \in H(\tau)$. Hence $N, \sigma_2, \tau_2 \vDash \Box_i\alpha$ and $N, \sigma_2, u_2 \vDash \alpha$ (since $S_i(\tau_2, u_2)$ holds).
- Case $u = w\langle\top\rangle_i$, $w = v\langle E\rangle_j$ with $j \neq i$, $r(w, w_2)$, and $S_i(w_2, u_2)$: Since $\alpha \in H(u)$, we have $\Box_i\alpha \in H(w)$. Hence, by the inductive assumption, $N, \sigma_2, w_2 \vDash \Box_i\alpha$. It follows that $N, \sigma_2, u_2 \vDash \alpha$, since $S_i(w_2, u_2)$ holds.
- Case $u = v\langle\top\rangle_i$, $w = v\langle E\rangle_i$, $r(w, w_2)$, and $S_i(w_2, u_2)$: By the assertion (*), there exists v_2 such that $r(v, v_2)$ and $S_i(v_2, w_2)$. Thus $S_i(v_2, u_2)$ holds. Since $\alpha \in H(u)$, we have $\Box_i\alpha \in H(v)$. By the inductive assumption, it follows that $N, \sigma_2, v_2 \vDash \Box_i\alpha$. Hence $N, \sigma_2, u_2 \vDash \alpha$.

Similarly as for the case of $BSMM$, it can be shown that if $r(w, w_2)$ and $R'_i(w, w\langle E\rangle_i)$ hold then $\sigma_2(w_2, \langle E\rangle_i)$ is defined. Therefore, the second condition in the above definition of r can be simplified to "if $r(w, w_2)$ and $R'_i(w, w\langle E\rangle_i)$ hold then $r(w\langle E\rangle_i, \sigma_2(w_2, \langle E\rangle_i))$".

To prove that $M \leq_r N$, there remains to show that if $r(w, w_2)$ and $R_i(w, u)$ then there exists $u_2 \in W_2$ such that $r(u, u_2)$ and $S_i(w_2, u_2)$. Suppose that $r(w, w_2)$ and $R_i(w, u)$ hold. We have that either $u = w\langle E\rangle_i$ for some E or there exists v such that $w = v\langle E\rangle_i$ and $u = v\langle F\rangle_i$ for some E, F. If the first case occurs then choose $u_2 = \sigma_2(w, \langle E\rangle_i)$. Consider the second case. Since $w = v\langle E\rangle_i$ and $r(w, w_2)$, by the assertion (*), there exists v_2 such that $r(v, v_2)$ and $S_i(v_2, w_2)$. Since $r(v, v_2)$ and $u = v\langle F\rangle_i$, there exists u_2 such that $r(u, u_2)$ and $S_i(v_2, u_2)$. Thus, we have $r(u, u_2)$ and $S_i(w_2, u_2)$. This completes the proof of that the standard L-model graph of I is a least L-model of I. •

Expected Lemma 3.3.2 *Let I be an L-normal model generator, M the standard L-model of I, and α a ground L-MProlog goal atom. Suppose that $M \vDash \alpha$. Then α is an L-instance of some atom of $Sat_L(I)$.*

Proof. Let $\langle W, \tau, R_1, \ldots, R_m, H\rangle$ be the standard L-model graph of I, $\boxdot = \Box_{i_1} \ldots \Box_{i_k}$, and $w = \langle\top\rangle_{i_1} \ldots \langle\top\rangle_{i_k}$. Suppose that α is of the form $\boxdot E$. Since $M \vDash \alpha$, we have $M, w \vDash E$. Hence, by Lemma 3.2.1, $\boxdot E \in Ext_L(I)$, and we also have $\boxdot E \in Sat_L(I)$. Now suppose that α is of the form $\boxdot\Diamond_i E$ with $i \neq i_k$. Since $M \vDash \alpha$, we have $M, w \vDash \Diamond_i E$ and there exists ∇_i such that $\nabla_i E \in H(w)$. Hence, by Lemma 3.2.1, $\boxdot\nabla_i E \in Ext_L(I)$. Therefore $\boxdot\Diamond_i E$ is an instance of some atom of $Sat_L(I)$. •

Expected Lemma 3.3.5 *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.*

Proof. Let M be the standard L-model of I and σ the standard $\Diamond$-realization function on M. By the definition of L-instances of program clauses and the construction of M, it is sufficient to prove that for any ground L-instance $\boxdot(A \leftarrow B_1, \ldots, B_n)$ of some program clause of P and for any $w \in W$ being an L-instance of $\boxdot$, $M, w \vDash (A \leftarrow B_1, \ldots, B_n)$. Suppose that $M, w \vDash B_i$ for all $1 \leq i \leq n$. We show that $M, w \vDash A$.

Let $\triangle = w = \langle E_1\rangle_{i_1} \ldots \langle E_k\rangle_{i_k}$. We first show that for any ground simple atom B of the form E, $\Box_i E$, or $\Diamond_i E$, if $M, w \vDash B$ then $\triangle B$ is an L-instance of some atom from $Sat_L(I)$. Suppose that $M, w \vDash B$. If $k = 0$ or $i_k \neq i$, then let $v = w$, else let $v = \langle E_1\rangle_{i_1} \ldots \langle E_{k-1}\rangle_{i_{k-1}}$.

- If $B = E$, then by Lemma 3.2.1, some $\Box$-lifting form of $\triangle B$ belongs to $Ext_L(I)$, and hence $\triangle B$ is an L-instance of some atom from $Sat_L(I)$.
- Now suppose that $B = \Box_i E$. Let $u = v\langle\top\rangle_i$ and $\triangle' = v\Box_i$. We have that $R_i(w, u)$, and hence $M, u \vDash E$. By Lemma 3.2.1, it follows that some $\Box$-lifting form of $\triangle' E$ belongs to $Ext_L(I)$. Hence, by the definition of Sat_L, $\triangle B$ is an L-instance of some atom from $Sat_L(I)$.
- Next, suppose that $B = \Diamond_i E$. Since $M, w \vDash B$, there exists F such that $v\langle F\rangle_i$ is a world of M and $M, v\langle F\rangle_i \vDash E$. Let $\triangle' = v\langle F\rangle_i$. By Lemma 3.2.1, some $\Box$-lifting form of $\triangle' E$ belongs to $Ext_L(I)$. Hence, by the definition of Sat_L, $\triangle B$ is an L-instance of some atom from $Sat_L(I)$.

Since $M, w \vDash B_i$ for $1 \leq i \leq n$, it follows that $\triangle B_i$ is an L-instance of some atom from $Sat_L(I)$. Consequently, $\triangle A$ is an L-instance of some atom α from $T_{0L,P}(Sat_L(I))$. Let α' be the L-normal form of α, i.e. $NF_L(\{\alpha\}) = \{\alpha'\}$. We have $\alpha' \in T_{L,P}(I) \subseteq I$. By Lemma 3.2.2, $M, \sigma \vDash \alpha'$. From this we can conclude that $M, w \vDash A$. •

4.4.3 A Schema for Semantics of $KD4I_g5_a$-MProlog

In this subsection, L denotes the logic $KD4I_g5_a$. In Table 4.9, we present an instantiation of our framework for $KD4I_g5_a$-MProlog. L-normal form of modalities and the rules (3), (4), (6), (7), (12), (13) are justified by the L-tautology $\nabla'_i\nabla_i\varphi \equiv \nabla_i\varphi$ with $g(i)$ being a singleton and ∇'_i and ∇_i being unlabeled modal operators. The rules (1) and (9) are based on axiom (I); the rules (2) and (11) are based on axiom (4); the rule (5) is based on axioms (4), (I), and (D); the rule (10) is based on axioms (I) and (D); and finally, the rule (14) is based on axioms (4) and (I).

Since the problem of checking satisfiability in the propositional version of $KD4I_g5_a$ is decidable [34], the problem of checking whether $\boxdot$ is an L-context instance of $\boxdot'$ is also decidable.

$L = KD4I_g5_a$, $\quad L$-MProlog

$\preceq_L$ is defined by Definition 3.3.1 at page 34.
A modality $\nabla_{i_1}^{(1)} \dots \nabla_{i_k}^{(k)}$ is in *L-normal form* if
for all $1 \leq j < k$ if $g(i_j)$ is a singleton then $i_j \neq i_{j+1}$.

Rules specifying operators Ext_L, Sat_L, NF_L, rNF_L, $rSat_L$:

Ext_L
$\triangle\Box_i\alpha \to \triangle\Box_j\alpha$ if $g(i) \supset g(j)$ (1)
$\triangle\Box_i\alpha \to \triangle\Box_i\Box_i\alpha$ (2)
$\triangle\nabla_i\Box_i\alpha \to \triangle\Box_i\alpha$ if $g(i)$ is a singleton (3)

Sat_L the rules specifying Ext_L plus
$\triangle\langle F\rangle_i E \to \triangle\Box_i\Diamond_i E$ if $g(i)$ is a singleton (4)
$\triangle\nabla\nabla' E \to \triangle\Diamond_i E$ if $\Diamond_i \preceq_L \nabla$ and $\Diamond_i \preceq_L \nabla'$ (5)

NF_L
$\triangle\nabla_i\nabla'_i E \to \triangle\nabla'_i E$ if $g(i)$ is a singleton and
∇'_i is of the form $\Box_i$ or $\langle E\rangle_i$ (6)

rNF_L
$\triangle\nabla_i E \leftarrow \triangle\langle X\rangle_i\nabla_i E$ if $g(i)$ is a singleton,
∇_i is of the form $\Box_i$ or $\langle E\rangle_i$, and X is a fresh atom variable (7)

$rSat_L$
$\triangle\Diamond_i E \leftarrow \triangle\langle X\rangle_i E$ for X being a fresh atom variable (8)
$\triangle\Diamond_i E \leftarrow \triangle\Diamond_j E$ if $g(i) \supset g(j)$ (9)
$\triangle\nabla_i\alpha \leftarrow \triangle\Box_j\alpha$ if $g(i) \subseteq g(j)$ (10)
$\triangle\Box_i\Box_i\alpha \leftarrow \triangle\Box_i\alpha$ (11)
$\triangle\Box_i\alpha \leftarrow \triangle\langle X\rangle_i\Box_i\alpha$ if $g(i)$ is a singleton and
X is a fresh atom variable (12)
$\triangle\nabla_i\Diamond_i E \leftarrow \triangle\Diamond_i E$ if $g(i)$ is a singleton (13)
$\triangle\Diamond_i E \leftarrow \triangle\langle X\rangle_j\Diamond_i E$ if $g(i) \supseteq g(j)$ and
X is a fresh atom variable (14)

Table 4.9: A schema for semantics of $KD4I_g5_a$-MProlog

Example 4.4.2 We give below an SLD-refutation of $P_{wise\_men} \cup \{\leftarrow \Box_a white(a)\}$ in $KD4I_g5_a$, where P_{wise_men} is the $KD4I_g5_a$-MProlog program given in Section 2.9. To increase readability, we recall a fragment of P_{wise_men}:

$$\begin{aligned}
\varphi_1 &= \Box_{abc}\,(white(a) \leftarrow \Diamond_b\, white(a))\\
\varphi_2 &= \Box_{abc}\,(white(a) \leftarrow \Diamond_c\, white(a))\\
\varphi_6 &= \Box_{abc}\,(\Box_c\, black(b) \leftarrow black(b))\\
\varphi_7 &= \Box_{abc}\,(white(a) \leftarrow black(b), black(c))
\end{aligned}$$

$\varphi_{10} = \Box_{abc}\Diamond_b\, black(b)$
$\varphi_{11} = \Box_{abc}\Diamond_c\, black(c)$

Here is an SLD-refutation of $P_{wise\_men} \cup \{\leftarrow \Box_a white(a)\}$ in $KD4I_g5_a$:

Goals	Input clauses/rules	MGUs
$\leftarrow \Box_a\, white(a)$		
$\leftarrow \Box_a\Diamond_b\, white(a)$	φ_1	
$\leftarrow \Box_a\langle X_2\rangle_b\, white(a)$	(8)	
$\leftarrow \Box_a\langle X_2\rangle_b\Diamond_c\, white(a)$	φ_2	
$\leftarrow \Box_a\langle X_2\rangle_b\langle X_4\rangle_c\, white(a)$	(8)	
$\leftarrow \Box_a\langle X_2\rangle_b\langle X_4\rangle_c\, black(b), \Box_a\langle X_2\rangle_b\langle X_4\rangle_c\, black(c)$	φ_7	
$\leftarrow \Box_a\langle X_2\rangle_b\, black(b), \Box_a\langle X_2\rangle_b\langle X_4\rangle_c\, black(c)$	φ_6	
$\leftarrow \Box_a\langle black(b)\rangle_b\langle X_4\rangle_c\, black(c)$	φ_{10}	$\{X_2/black(b)\}$
$\diamond$	φ_{11}	$\{X_4/black(c)\}$

Theorem 4.4.3 *The schema given in Table 4.9 for semantics of $KD4I_g5_a$-MProlog is correct.*

To prove this theorem we have to prove Expected Theorem 3.2.3 and Expected Lemmas 3.2.2, 3.3.2, 3.3.4, 3.3.5, 3.5.1 – 3.5.4.

Expected Lemma 3.3.1 (page 35) clearly holds. The proof of Expected Lemma 3.3.4 is the same as for $KDI45$ (see page 67).

Expected Lemma 3.2.2 *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 $M, \sigma \vDash I$.*

Proof. By the definition, M is an L-model. Let $(R'_i)_{1\le i\le m}$ be the skeleton of M. We prove by induction on the length of α that for any $w \in W$, if $\alpha \in H(w)$ then $M, \sigma, w \vDash \alpha$. The cases when α is a classical atom or $\alpha = \langle E\rangle_i\beta$ are trivial. Consider the remaining case when $\alpha = \Box_i\beta$. Let u be a world such that $R_i(w,u)$ holds. We show that $\beta \in H(u)$ by induction on the derivation of $R_i(w,u)$:

- The case $R'_i(w,u)$ holds is trivial.
- Case $R_i(w,u)$ is derived from $R_j(w,u)$ with $j < i$: Treating w as a modality, by Lemma 3.2.1, a $\Box$-lifting form of $w\Box_i\beta$ belongs to $Ext_L(I)$. By the definition of Ext_L, a $\Box$-lifting form of $w\Box_j\beta$ belongs to $Ext_L(I)$. By Lemma 3.2.1, it follows that $\Box_j\beta \in H(w)$. Hence, by the inductive assumption, $\beta \in H(u)$.
- Case $R_i(w,u)$ is derived from $R_j(w,v)$ and $R_k(v,u)$ with $j \le i$ and $k \le i$: By Lemma 3.2.1, a $\Box$-lifting form of $w\Box_i\beta$ belongs to $Ext_L(I)$. By the definition of Ext_L, a $\Box$-lifting form of $w\Box_j\Box_k\beta$ belongs to $Ext_L(I)$. By Lemma 3.2.1, it follows that $\Box_j\Box_k\beta \in H(w)$. Hence, by the inductive assumption, $\Box_k\beta \in H(v)$ and $\beta \in H(u)$.
- Case $g(i)$ is a singleton and $R_i(w,u)$ is derived from $R'_i(v,w)$ and $R'_k(v,u)$: By Lemma 3.2.1, a $\Box$-lifting form of some $v\nabla_i\Box_i\beta$ belongs to $Ext_L(I)$. By the definition of Ext_L, a $\Box$-lifting form of $v\Box_i\beta$ belongs to $Ext_L(I)$. By Lemma 3.2.1, it follows that $\Box_i\beta \in H(v)$. Hence, by the inductive assumption, $\beta \in H(u)$.

•

Expected Theorem 3.2.3 *The standard L-model of an L-normal model generator I is a least L-model of I.*

Proof. Let $M = \langle W, \tau, R_1, \ldots, R_m, H\rangle$ be the standard L-model graph of I, σ the standard $\Diamond$-realization function and $(R'_i)_{1\leq i\leq m}$ the skeleton of the standard L-model of I. By Lemma 3.2.2, M is an L-model of I. Let $N = \langle D, W_2, \tau_2, S_1, \ldots, S_m, \pi\rangle$ be an arbitrary L-model of I and σ_2 a $\Diamond$-realization function on N such that $N, \sigma_2 \vDash I \cup \{\boxdot\langle\top\rangle_i \mid \boxdot \text{ is a universal modality and } 1 \leq i \leq m\}$.

Let $r \subseteq W \times W_2$ be the least relation such that, for all w, w_2, u_2, E, i :

- $r(\tau, \tau_2)$;
- if $r(w, w_2)$ and $R'_i(w, w\langle E\rangle_i)$ hold and $\sigma_2(w_2, \langle E\rangle_i)$ is defined then $r(w\langle E\rangle_i, \sigma_2(w_2, \langle E\rangle_i))$;
- if $r(w, w_2)$ and $S_i(w_2, u_2)$ hold then $r(u, u_2)$ holds for u such that:
  - if $w = \tau$ then $u = \langle\top\rangle_i$,
  - if $w = v\langle E\rangle_j$ and ($j \neq i$ or $g(i)$ is not a singleton) then $u = w\langle\top\rangle_i$,
  - and if $w = v\langle E\rangle_i$ and $g(i)$ is a singleton then $u = v\langle\top\rangle_i$.

Analogously as for the case of $KD45_{(m)}$ (see page 72), it can be shown that $M \leq_r N$. •

Expected Lemma 3.3.2 *Let I be an L-normal model generator, M the standard L-model of I, and α a ground L-MProlog goal atom. Suppose that $M \vDash \alpha$. Then α is an L-instance of some atom of $Sat_L(I)$.*

Proof. Let $\langle W, \tau, R_1, \ldots, R_m, H\rangle$ be the standard L-model graph of I, $\boxdot = \Box_{i_1} \ldots \Box_{i_k}$ be a modality, and $w = \langle\top\rangle_{i_1} \ldots \langle\top\rangle_{i_k}$.

- Consider the case α is of the form $\boxdot E$. Since $M \vDash \alpha$, we have $M, w \vDash E$. Hence, by Lemma 3.2.1, $\boxdot E \in Ext_L(I)$, and we also have $\boxdot E \in Sat_L(I)$.
- Now consider the case α is of the form $\boxdot\Diamond_i E$ with the property that if $g(i)$ is a singleton then $i \neq i_k$. Since $M \vDash \alpha$, we have $M, w \vDash \Diamond_i E$. Hence there exists u such that $R_i(w, u)$ holds and $E \in H(u)$.

  It can be shown that u is of the form $w\langle F_1\rangle_{j_1} \ldots \langle F_h\rangle_{j_h}$ with $h \geq 1$ and $g(l) \subseteq g(i)$ for all $l \in \{j_1, \ldots, j_h\}$. (First, by induction on the derivation of $R_i(w, u)$, it can be shown that there are possible worlds $w_0, \ldots, w_n$ and indices $s_1, \ldots, s_n$ such that $w_0 = w$, $w_n = u$, and for every $1 \leq t \leq n$, either $R'_{s_t}(w_{t-1}, w_t)$ holds or $R_{s_t}(w_{t-1}, w_t)$ holds and $g(s_t)$ is a singleton. Observe that if $R_s(v, v')$ and $R_s(v', v'')$ hold then $R_s(v, v'')$ holds; if $R'_j(v, v')$ and $R_s(v', v'')$ hold, $j \neq s$, and $g(s)$ is a singleton, then $R'_s(v', v'')$ holds. Hence, we can assume that $R'_{s_t}(w_{t-1}, w_t)$ holds for all $1 \leq t \leq n$. Take $h = n$ and $j_t = s_t$ for $1 \leq t \leq n$.)

  By Lemma 3.2.1, some $\Box$-lifting form of $w\langle F_1\rangle_{j_1} \ldots \langle F_h\rangle_{j_h} E$ belongs to $Ext_L(I)$. It follows that some $\Box$-lifting form of $\boxdot\langle F_1\rangle_{j_1} \ldots \langle F_h\rangle_{j_h} E$ belongs to $Ext_L(I)$ and $Sat_L(I)$. Hence $\boxdot\Diamond_i E$ is an L-instance of some atom from $Sat_L(I)$.

•

Expected Lemma 3.3.5 *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.*

Proof. Let M be the standard L-model of I and σ the standard $\Diamond$-realization function on M. By the definition of L-instances of program clauses and the construction of M, it is sufficient to prove that for any ground L-instance $\boxdot(A \leftarrow B_1, \ldots, B_n)$ of some program clause of P, for any $w \in W$ being an L-instance of $\boxdot$, $M, w \vDash (A \leftarrow B_1, \ldots, B_n)$. Suppose that $M, w \vDash B_i$ for all $1 \leq i \leq n$. We show that $M, w \vDash A$.

Let $\triangle = w = \langle E_1\rangle_{i_1} \ldots \langle E_k\rangle_{i_k}$. We first show that for any ground simple atom B of the form E, $\Box_i E$, or $\Diamond_i E$, if $M, w \vDash B$ then $\triangle B$ is an L-instance of some atom from $Sat_L(I)$. Suppose that $M, w \vDash B$. If $k \geq 1$ and $i = i_k$ and $g(i)$ is a singleton, then let $v = \langle E_1\rangle_{i_1} \ldots \langle E_{k-1}\rangle_{i_{k-1}}$, else let $v = w$.

If $B = E$, then by Lemma 3.2.1, some $\Box$-lifting form of $\triangle B$ belongs to $Ext_L(I)$, and hence $\triangle B$ is an L-instance of some atom from $Sat_L(I)$.

Now suppose that $B = \Box_i E$. Let $u = v\langle\top\rangle_i$ and $\triangle' = v\Box_i$. We have $R_i(w, u)$, and hence $M, u \vDash E$. By Lemma 3.2.1, it follows that some $\Box$-lifting form of $\triangle' E$ belongs to $Ext_L(I)$. Hence, $\triangle B$ is an L-instance of some atom from $Sat_L(I)$.

Next, suppose that $B = \Diamond_i E$. Consider the case $w \neq v$ (i.e. $i = i_k$ and $g(i)$ is a singleton). Since $M, w \vDash B$, there exists F such that $v\langle F\rangle_i$ is a world of M and $M, v\langle F\rangle_i \vDash E$. Let $\triangle' = v\langle F\rangle_i$. By Lemma 3.2.1, some $\Box$-lifting form of $\triangle' E$ belongs to $Ext_L(I)$. Hence, by the rules (2) and (4) of Sat_L, $\triangle B$ is an L-instance of some atom from $Sat_L(I)$. Now consider the case $w = v$ (i.e. $k = 0$ or $i \neq i_k$ or $g(i)$ is not a singleton). Since $M, w \vDash \Diamond_i E$, as shown in the proof of Expected Lemma 3.3.2, there exists $u = w\langle F_1\rangle_{j_1} \ldots \langle F_h\rangle_{j_h}$ such that $M, u \vDash E$, $h \geq 1$, and $g(l) \subseteq g(i)$ for all $l \in \{j_1, \ldots, j_h\}$. By Lemma 3.2.1, some $\Box$-lifting form of $w\langle F_1\rangle_{j_1} \ldots \langle F_h\rangle_{j_h} E$ belongs to $Ext_L(I)$. It follows that some $\Box$-lifting form of $\triangle\langle F_1\rangle_{j_1} \ldots \langle F_h\rangle_{j_h} E$ belongs to $Sat_L(I)$. By the rules of Sat_L, some $\Box$-lifting form of $\triangle\Diamond_i E$ belongs to $Sat_L(I)$. Hence $\triangle B$ is an L-instance of some atom from $Sat_L(I)$.

Since $M, w \vDash B_i$ for $1 \leq i \leq n$, it follows that $\triangle B_i$ is an L-instance of some atom from $Sat_L(I)$. Consequently, $\triangle A$ is an L-instance of some atom α from $T_{0L,P}(Sat_L(I))$. Let α' be the L-normal form of α, i.e. $NF_L(\{\alpha\}) = \{\alpha'\}$. We have $\alpha' \in T_{L,P}(I) \subseteq I$. By Lemma 3.2.2, $M, \sigma \vDash \alpha'$. If $\alpha' = \alpha$ then we can derive from $M, \sigma \vDash \alpha'$ that $M, w \vDash A$. Suppose that $\alpha' \neq \alpha$. Thus, α is of the form $\triangle''\nabla_i\nabla'_i E$, where $\triangle''\nabla_i = \triangle$, $g(i)$ is a singleton, and ∇'_i is $\Box_i$ or $\langle E\rangle_i$. If $\nabla'_i = \langle E\rangle_i$ then $A = \Diamond_i E$. We have that $\alpha' = \triangle''\nabla'_i E$. Since $M, \sigma \vDash \alpha'$ and $g(i)$ is a singleton, it follows that $M, \sigma \vDash \triangle''\Box_i A$. Hence $M, w \vDash A$. This completes the proof. •

Expected Lemma 3.5.1 *Let $\triangle$ and $\triangle'$ be ground modalities in L-normal labeled form. Let B be an atom of the form E, $\Diamond_i E$, or $\Box_i E$, and B' an atom of the form E, $\Diamond_j E$, $\langle X\rangle_j E$, or $\Box_j E$, where X is a fresh atom variable. Suppose that $\triangle$ is an L-instance of $\triangle'$ and B is an L-instance of B'. Then $\triangle' B'$ is derivable from $\triangle B$ using $rSat_L$.*

Proof. Since $\triangle$ is a ground modality and is an L-instance of $\triangle'$, $\triangle' B$ is derivable from $\triangle B$ using the rule (10). Since B is an L-instance of B', $\triangle' B'$ is derivable from $\triangle' B$ using the rules (8), (9), and (10). Hence $\triangle' B'$ is derivable from $\triangle B$ using $rSat_L$. •

Expected Lemma 3.5.2 *Suppose that β is an atom in almost L-normal labeled form and $\alpha \in Sat_L(\{\beta\})$ or $\alpha \in NF_L(\{\beta\})$. Then there exists an atom β' and a substitution θ s.t. $\beta = \beta'\theta$, the domain of θ consists of fresh atom variables, and β' is derivable from α using $rSat_L$ and rNF_L.*

Proof. It is sufficient to consider the case when α is derived from β using an application of one Sat_L/NF_L rule. We give here a proof only for one representative case, when the used rule is $\triangle\langle F\rangle_i E \to \triangle\Box_i\Diamond_i E$, where $g(i)$ is a singleton. Suppose that $\alpha = \triangle\Box_i\Diamond_i E$ and $\beta = \triangle\langle F\rangle_i E$. By applying the $rSat_L$ rule (13) to α, we can derive $\triangle\Diamond_i E$. By applying the $rSat_L$ rule (8) to $\triangle\Diamond_i E$, we can derive $\triangle\langle X\rangle_i E$, where X is a fresh atom variable. Choose $\beta' = \triangle\langle X\rangle_i E$. We have that $\beta = \beta'\{X/F\}$ and β' is derivable from α using $rSat_L$. •

Expected Lemma 3.5.3 *Let $\beta = rSat_L(\alpha)$, M be an L-model, σ a $\Diamond$-realization function on M, and θ a substitution. Suppose that $M, \sigma \vDash \forall_c(\beta'\theta)$ for some $\Box$-lifting form β' of β. Then $M, \sigma \vDash \forall_c(\alpha'\theta)$ for some $\Box$-lifting form α' of α.*

Proof. There are the following cases to consider:

- Case when the rule (8) is used, $\alpha = \triangle\Diamond_i E$ and $\beta' = \triangle'\nabla_i E$: Choose $\alpha' = \triangle'\Diamond_i E$.
- Case when the rule (9) is used, $\alpha = \triangle\Diamond_i E$ and $\beta' = \triangle'\nabla_j E$: Choose $\alpha' = \triangle'\Diamond_i E$.
- Case when the rule (10) is used, $\alpha = \triangle\nabla_i\gamma$ and $\beta' = \triangle'\Box_j\gamma'$: Choose $\alpha' = \triangle'\Box_i\gamma'$.
- Case when the rule (11) is used, $\alpha = \triangle\Box_i\Box_i\gamma$ and $\beta' = \triangle'\Box_i\gamma'$: Choose $\alpha' = \triangle'\Box_i\Box_i\gamma'$.
- Case when the rule (12) is used, $\alpha = \triangle\Box_i\gamma$ and $\beta' = \triangle'\nabla_i\Box_i\gamma'$: Choose $\alpha' = \triangle'\Box_i\gamma'$.
- Case when the rule (13) is used, $\alpha = \triangle\nabla_i\Diamond_i E$ and $\beta' = \triangle'\nabla_i E$: Choose $\alpha' = \triangle'\Box_i\Diamond_i E$.
- Case when the rule (14) is used, $\alpha = \triangle\Diamond_i E$ and $\beta' = \triangle'\nabla_j\nabla'_i E$: Choose $\alpha' = \triangle'\Diamond_i E$.

Note that α' is a $\Box$-lifting form of α. It is easy to see that, since $M, \sigma \vDash \forall_c(\beta'\theta)$ and the condition of the used rule holds, $M, \sigma \vDash \forall_c(\alpha'\theta)$. •

Expected Lemma 3.5.4 *Let $\beta =_\delta rNF_L(\alpha)$, M be an L-model, σ a maximal $\Diamond$-realization function on M, and θ a substitution. Suppose that $M, \sigma \vDash \forall_c(\beta'\theta)$ for some $\Box$-lifting form β' of β. Then $M, \sigma \vDash \forall_c(\alpha'\delta\theta)$ for some $\Box$-lifting form α' of α.*

Proof. There is only one rNF_L rule. Let $\alpha\delta = \triangle\nabla_i E$ and $\beta = \triangle\langle X\rangle_i\nabla_i E$, where $g(i)$ is a singleton, ∇_i is $\Box_i$ or $\langle E\rangle_i$, and X is a fresh atom variable. Let $\beta' = \triangle'\nabla'_i\nabla''_i E$. We have that ∇''_i is a $\Box$-lifting form of ∇_i. Since $M, \sigma \vDash \forall_c(\beta'\theta)$ and σ is a maximal $\Diamond$-realization function on M, it follows that $M, \sigma \vDash \forall_c(\triangle'\nabla_i E\theta)$. Since $\triangle'$ is $\Box$-lifting of $\triangle$, there exists a $\Box$-lifting form α' of α such that $\alpha'\delta = \triangle'\nabla_i E$, and we have $M, \sigma \vDash \forall_c(\alpha'\delta\theta)$. •

4.5 Semantics of MProlog in Basic Serial Monomodal Logics

In this section, using our framework we specify semantics for MProlog in the basic serial monomodal logics *KD*, *T*, *KDB*, *B*, *KD*4, *S*4, *KD*5, *KD*45, *S*5, which were published in our paper [55]. Let L denote one of these logics.

In monomodal logics, we have that $m = 1$, and $\Box_1$, $\Diamond_1$, $\langle E\rangle_1$, $\langle X\rangle_1$, R_1 are denoted respectively by $\Box$, $\Diamond$, $\langle E\rangle$, $\langle X\rangle$, R. A modality consisting of h modal operators $\Box$ is denoted by $\Box^h$.

A modality is in L-normal form for $L \in \{KD45, S5\}$ if its length is 1 or 0. A modality is in *KD*5-normal form if it is of the form ∇, $\Box\nabla$, or empty (where ∇ denotes a modal operator).

For L being one of the remaining logics KD, T, KDB, B, $KD4$, $S4$, no restrictions are adopted for L-normal form of modalities.

Ordering between modal operators is defined as usual, i.e. $\Diamond \preceq \langle E \rangle \preceq \Box$, $\Diamond \preceq \langle X \rangle \preceq \Box$.

It is easily seen that a universal modality $\Box^k$ in L-normal form is an L-context instance of a universal modality $\Box^h$ in L-normal form iff:

- case $L = KD : k = h$,
- case $L = T : k \leq h$,
- case $L = KDB : k = h - 2l$, for some $l \geq 0$,
- case $L = B : k \leq h$,
- case $L = KD4 : k \geq h > 0$ or $k = h = 0$,
- case $L = S4 : k = 0$ or ($k > 0$ and $h > 0$),
- case $L = KD5 : k = h \in \{0, 1, 2\}$ or ($k = 1$ and $h = 2$),
- case $L = KD45 : k = h \in \{0, 1\}$,
- case $L = S5 : k = 0$ or $k = h = 1$.

The operator Ext_L is specified by the following rules:

- $\triangle\Box\alpha \rightarrow \triangle\alpha$ if $L \in \{T, B, S4\}$,
- $\triangle\nabla\Box\alpha \rightarrow \triangle\alpha$ if $L \in \{KDB, B\}$,
- $\triangle\Box\alpha \rightarrow \triangle\Box\Box\alpha$ if $L \in \{KD4, S4\}$,
- $\Box\Box E \rightarrow \Box E$ if $L = KD5$,
- $\Box E \rightarrow E$ if $L = S5$,
- no rules for $L \in \{KD, KD45\}$.

The operator Sat_L is specified by the rules specifying Ext_L and the following ones, whose right hand side atoms contain no more than one $\Diamond$:

- $\triangle E \rightarrow \triangle\Diamond E$ if $L \in \{T, B, S4\}$,
- $\triangle E \rightarrow \triangle\Box\Diamond E$ if $L \in \{KDB, B\}$,
- $\triangle\nabla\nabla' E \rightarrow \triangle\Diamond E$ if $L \in \{KD4, S4\}$,
- $\Box\Box E \rightarrow \Box\Box\Box E$ if $L = KD5$,
- $\Box E \rightarrow \Box\Box E$ if $L \in \{KD45, S5\}$,
- $\nabla E \rightarrow \Box\Diamond E$ if $L \in \{KD5, KD45, S5\}$,
- $\Box\nabla E \rightarrow \Box\Box\Diamond E$ if $L = KD5$,
- $E \rightarrow \Diamond E$ if $L = S5$,
- no rules for $L = KD$.

The operator NF_L is specified by the following rules:

- $\nabla\nabla' E \to \nabla' E$ if $L \in \{KD45, S5\}$ and ∇' is either $\Box$ or $\langle E\rangle$,
- $\Box\nabla\nabla' E \to \Box\nabla' E$ if $L = KD5$ and ∇' is either $\Box$ or $\langle E\rangle$,
- $\nabla\nabla' \to \Box\nabla'$ if $L = KD5$,
- no rules for $L \in \{KD, T, KDB, B, KD4, S4\}$.

The operator $rSat_L$ is specified by the following rules, in which X is a fresh atom variable:

- $\triangle\Diamond E \leftarrow \triangle\langle X\rangle E$ (the backward labeling rule),
- $\triangle\nabla\alpha \leftarrow \triangle\Box\alpha$ (the $\Box$-lifting rule),
- if $L \in \{T, B, S4\}$: $\triangle\alpha \leftarrow \triangle\Box\alpha$ and $\triangle\Diamond E \leftarrow \triangle E$,
- if $L \in \{KDB, B\}$: $\triangle\alpha \leftarrow \triangle\langle X\rangle\Box\alpha$ and $\triangle\Box\Diamond E \leftarrow \triangle E$,
- if $L \in \{KD4, S4\}$: $\triangle\Box\Box\alpha \leftarrow \triangle\Box\alpha$ and $\triangle\Diamond E \leftarrow \triangle\langle X\rangle\Diamond E$,
- if $L \in \{KD5, KD45, S5\}$: $\Box\Diamond E \leftarrow \Diamond E$,
- if $L = KD5$: $\Box E \leftarrow \Box\Box E$ and $\Box\Box\Box E \leftarrow \Box\Box E$ and $\Box\Box\Diamond E \leftarrow \Box\Diamond E$,
- if $L \in \{KD45, S5\}$: $\Box\Box E \leftarrow \Box E$,
- if $L = S5$: $E \leftarrow \Box E$ and $\Diamond E \leftarrow E$.

In [55], the backward labeled rule and the $\Box$-lifting rule stand alone, while in this work we clasify them as rules specifying $rSat_L$. We also make a slight reformulation of $rSat_L$ rules for $L \in \{KD5, KD45, S5\}$, which repairs a gap in [55] for the case of $S5$.

The operator rNF_L is specified by the following rules, in which X is a fresh atom variable:

- $\nabla E \leftarrow \langle X\rangle\nabla E$ if $L \in \{KD45, S5\}$ and ∇ is either $\Box$ or $\langle E\rangle$,
- $\Box\nabla E \leftarrow \Box\langle X\rangle\nabla E$ if $L = KD5$ and ∇ is either $\Box$ or $\langle E\rangle$,
- $\Box\nabla E \leftarrow \langle X\rangle\nabla E$ if $L = KD5$,
- no rules for $L \in \{KD, T, KDB, B, KD4, S4\}$.

In [55], for the first two rules defining rNF_L, we do not require ∇ to be $\Box$ or $\langle E\rangle$. This difference is not essential for soundness and completeness of the SLD-resolution calculus. For example, having $\leftarrow \langle X\rangle\langle F\rangle E$ (resp. $\leftarrow \langle X\rangle\langle Y\rangle E$) in $L \in \{KD45, S5\}$, the only way to proceed is to resolve this goal with a program clause, which results in a unification of F (resp. Y) with E. Without the mentioned condition, however, Expected Lemma 3.5.4 does not hold for $L \in \{KD5, KD45, S5\}$.

Observe that our instantiation of the framework for $KD4$-MProlog coincides with the schema for semantics of $KDI4$-MProlog given in Table 4.4, and our instantiation for $KD45$-MProlog is very similar to the schema for semantics of $KD45_{(m)}$-MProlog given in Table 4.8.

Theorems 3.2.3, 3.3.6, 3.5.7, 3.5.15 for L being one of the basic serial monomodal logics are proved in [55]. (It can be shown that all the Expected Lemmas hold for our instantiation of the framework for MProlog in the basic serial monomodal logics.)

Chapter 5

Optimizations

In this chapter, we provide some optimizations for the framework and the given schemata for semantics of L-MProlog. In the first section, we present the standard form for SLD-refutations. In the second section, we modify our framework in some aspects to allow more efficient sets of rules for specifying operators Sat_L and $rSat_L$. We also optimize the given schemata for semantics of L-MProlog. In the third section, we study the version L-MProlog-$\Box$ of L-MProlog that disallows existential modal operators in programs and goals. We show that the schemata given for semantics of L-MProlog can be significantly simplified for L-MProlog-$\Box$. In the two last sections, we briefly discuss restrictions, iterative deepening search and tabulation, which can be used to control the search space for MProlog.

5.1 The Standard Form for Resolution Cycles

Definition 5.1.1 A *resolution cycle* is a fragment of an SLD-derivation that starts either immediately after an application of a program clause or from the beginning of the derivation, and ends with an application of a program clause.

Note that an SLD-refutation can be divided into a sequence of resolution cycles.

Definition 5.1.2 A selection rule is *standard* if in every resolution cycle only atoms at the same position are selected.

A resolution cycle in an SLD-derivation via a standard selection function is thus an application of a sequence of $rSat_L/rNF_L$ rules with a program clause at the end to a selected atom. (By an application of a rule/program clause we mean an application of its variant.)

Definition 5.1.3 A resolution cycle is in the *standard form* if it is an application of a sequence of $rSat_L$ rules, a sequence of rNF_L rules, and a program clause in this order to a selected atom.

In the following, we show that without loss of completeness we can adopt the restriction that every resolution cycle is in the standard form.

Observe that if we change Expected Lemma 3.5.2 from

> *Suppose that β is an atom in almost L-normal labeled form and $\alpha \in Sat_L(\{\beta\})$ or $\alpha \in NF_L(\{\beta\})$. Then there exists an atom β' and a substitution θ s.t. $\beta = \beta'\theta$, the domain of θ consists of fresh atom variables, and β' is derivable from α using $rSat_L$ and rNF_L.*

to

> *Suppose that β is an atom in almost L-normal labeled form and $\alpha \in Sat_L(\{\beta\})$ (resp. $\alpha \in NF_L(\{\beta\})$). Then there exists an atom β' and a substitution θ s.t. $\beta = \beta'\theta$, the domain of θ consists of fresh atom variables, and β' is derivable from α using $rSat_L$ (resp. using rNF_L and the rule $\triangle\nabla_i\nabla' E \leftarrow \triangle\Box_i\nabla' E$, which is an instance of an $rSat_L$ rule).*

then the lemma still holds for every modal logic L considered in this work.

With this change in mind, by reconsidering the completeness proof given in Section 3.5.3, it can be seen that we can assume that after applying an rNF_L rule to a goal atom, an rNF_L rule or the rule $\triangle\nabla_i\nabla' E \leftarrow \triangle\Box_i\nabla' E$ (which is an instance of an $rSat_L$ rule) or a program clause is applied to the obtained atom. Observe that the applications of the rule $\triangle\nabla_i\nabla' E \leftarrow \triangle\Box_i\nabla' E$ can be simulated at the beginning of the next resolution cycle. Hence, we can assume that after applying an rNF_L rule to a goal atom, an rNF_L rule or a program clause is applied to the obtained atom. (Note that, for any modal logic L considered in this work except $KDI45$, one application of an rNF_L rule is enough for one resolution cycle.) We arrive at:

Theorem 5.1.1 *For L being one of the considered modal logics, the SLD-resolution calculus for L-MProlog is still strongly complete (in the sense of Theorem 3.5.16) if we adopt the restriction that every resolution cycle is in the standard form (and the selection rule is standard).*

5.2 Optimizing the Framework and Its Instantiations

In the framework given for MProlog, we put Expected Lemma 3.2.2, which states that the standard L-model of an L-normal model generator I is an L-model of I. This assertion requires an appropriate set of rules for specifying Ext_L; for example, for $L = KDI4_s$, the rules are

$$\triangle\Box_i\alpha \rightarrow \triangle\Box_j\alpha \text{ if } i > j,$$
$$\triangle\Box_i\alpha \rightarrow \triangle\Box_j\Box_i\alpha.$$

Expected Lemma 3.2.2 is used to proved Expected Theorem 3.2.3, which states that the standard L-model of an L-normal model generator I is a least L-model of I. Then for all instantiations of the framework given in the previous chapter, we arranged that each rule for specifying Ext_L is also a rule for specifying Sat_L. This property was used in the proofs of Expected Lemmas 3.3.2 and 3.3.5. The set of rules specifying Sat_L in turn affects the set of rules specifying $rSat_L$. The relation between them is formulated in Expected Lemma 3.5.2.

We have an observation that, e.g. for $L = KDI4_s$, if we replace the rules specifying Ext_L (and the corresponding ones in the set of rules specifying Sat_L) by the following more restricted ones:

$$\triangle\Box_i E \rightarrow \triangle\Box_j E \text{ if } i > j$$
$$\triangle\Box_i E \rightarrow \triangle\Box_j\Box_i E$$

then, despite that Expected Lemma 3.2.2 and Expected Theorem 3.2.3 do not hold anymore[1], Theorem 3.3.6 (on correctness of the fixpoint semantics) still holds. (Recall that α stands for an atom, while E stands for a classical atom.) The intuition behind such an optimization is that: when constructing a least L-model for P and realizing a formula of the form $(A \leftarrow B_1, \ldots, B_n)$

[1]It can be shown that Expected Lemma 3.2.2 and Expected Theorem 3.2.3 still hold for the case when the considered model generator is $I_{L,P}$ for some L-MProlog program P.

$L = KDI4_s$, L-MProlog	
Ext_L	$\triangle\Box_i E \rightarrow \triangle\Box_j E$ if $i > j$ $\triangle\Box_i E \rightarrow \triangle\Box_j\Box_i E$
Sat_L	the rules specifying Ext_L plus $\triangle\nabla\nabla' E \rightarrow \triangle\Diamond_i E$ if $\Diamond_i \preceq_L \nabla'$
$rSat_L$	$\triangle\Diamond_i E \leftarrow \triangle\langle X\rangle_i E$ $\triangle\Diamond_i E \leftarrow \triangle\Diamond_j E$ if $i > j$ $\triangle\nabla\nabla'_i E \leftarrow \triangle\Box_i E$ $\triangle\Diamond_i E \leftarrow \triangle\langle X\rangle_j\Diamond_i E$

$L = KDI4$, L-MProlog	
Ext_L	$\triangle\Box_i E \rightarrow \triangle\Box_j E$ if $i > j$ $\triangle\Box_i E \rightarrow \triangle\Box_i\Box_i E$
Sat_L	the rules specifying Ext_L plus $\triangle\nabla\nabla' E \rightarrow \triangle\Diamond_i E$ if $\Diamond_i \preceq_L \nabla$ and $\Diamond_i \preceq_L \nabla'$
$rSat_L$	$\triangle\Diamond_i E \leftarrow \triangle\langle X\rangle_i E$ $\triangle\Diamond_i E \leftarrow \triangle\Diamond_j E$ if $i > j$ $\triangle\nabla_i\nabla_j E \leftarrow \triangle\Box_k E$ if $k = max(i, j)$ $\triangle\Diamond_i E \leftarrow \triangle\langle X\rangle_j\Diamond_i E$ if $i \geq j$

Table 5.1: Optimized sets of rules for L-MProlog

at a possible world w, to check whether $B_1, \ldots, B_n$ are true at w, only rules that manipulate *suffixes* of the modal context of an atom are needed. This does not hold for the whole classes $BSMM$ and $sCFG$ but is true for all the other concrete modal logics considered in this work.

Such a mentioned optimization for the set of rules specifying Sat_L makes a corresponding optimization for the set of rules specifying $rSat_L$. For example, for $L = KDI4_s$, the rules

$$\triangle\nabla_i\alpha \leftarrow \triangle\Box_j\alpha \text{ if } i \leq j,$$
$$\triangle\nabla\Box_i\alpha \leftarrow \triangle\Box_i\alpha,$$

can be replaced by

$$\triangle\nabla_i E \leftarrow \triangle\Box_j E \text{ if } i \leq j,$$
$$\triangle\nabla\Box_i E \leftarrow \triangle\Box_i E.$$

That is, by giving up Expected Lemma 3.2.2 and Expected Theorem 3.2.3 (on the standard L-model of an L-normal model generator) we are able to optimize both the sets of rules specifying Sat_L and $rSat_L$, which results in more efficient fixpoint semantics and SLD-resolution calculi for L-MProlog while preserving their correctness and completeness.

In the previous chapter, we have stated optimizations for the schemata for semantics of L-MProlog for $L \in \{KDI4_s5, KD4_s5_s, KD45_{(m)}\}$. The schemata for semantics of L-MProlog

$L = KDI45$, L-MProlog
The following rules are accompanied with the condition that the atoms in both sides are in L-normal labeled form for rules specifying Ext_L and in almost L-normal labeled form for the other rules. (*)
Ext_L $\triangle\Box_i E \to \triangle\Box_j E$ if $i > j$ $\triangle\Box_i E \to \triangle\Box_i\Box_j E$ if $i > j$ $\triangle\Box_i\Box_j E \to \triangle\Box_j E$ if $i > j$
Sat_L the rules for Ext_L with the modification stated in (*), plus $\triangle\Box_i E \to \triangle\Box_i\Box_i E$ $\triangle\nabla E \to \triangle\Box_i\Diamond_i E$ 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 $i > j$
NF_L $\triangle\nabla_i\nabla'_j E \to \triangle\nabla'_j E$ if $i \leq j$
rNF_L $\triangle\nabla_j E \leftarrow \triangle\langle X\rangle_i\nabla_j E$ if $i \leq j$
$rSat_L$ $\triangle\Diamond_i E \leftarrow \triangle\langle X\rangle_i E$ $\triangle\Diamond_i E \leftarrow \triangle\Diamond_j E$ if $i > j$ $\triangle\nabla_i\nabla_j E \leftarrow \triangle\Box_k E$ if $k = max(i, j)$ $\triangle\nabla_i E \leftarrow \triangle\Box_j\Box_i E$ if $i < j$ $\triangle\nabla_j\Diamond_i E \leftarrow \triangle\Diamond_i E$ if $j \leq i$ $\triangle\Diamond_i E \leftarrow \triangle\langle X\rangle_i\Diamond_i E$

Table 5.2: Optimized sets of rules for L-MProlog, c.d.

$L = KD4I_g5_a$, L-MProlog	
Ext_L	$\triangle\Box_i E \rightarrow \triangle\Box_j E$ if $g(i) \supset g(j)$ $\triangle\Box_i E \rightarrow \triangle\Box_i\Box_i E$ $\triangle\nabla_i\Box_i E \rightarrow \triangle\Box_i E$ if $g(i)$ is a singleton
Sat_L	the rules specifying Ext_L plus $\triangle\langle F\rangle_i E \rightarrow \triangle\Box_i\Diamond_i E$ if $g(i)$ is a singleton $\triangle\nabla\nabla' E \rightarrow \triangle\Diamond_i E$ if $\Diamond_i \preceq_L \nabla$ and $\Diamond_i \preceq_L \nabla'$
NF_L	$\triangle\nabla_i\nabla'_i E \rightarrow \triangle\nabla'_i E$ if $g(i)$ is a singleton
rNF_L	$\triangle\nabla_i E \leftarrow \triangle\langle X\rangle_i\nabla_i E$ if $g(i)$ is a singleton
$rSat_L$	$\triangle\Diamond_i E \leftarrow \triangle\langle X\rangle_i E$ $\triangle\Diamond_i E \leftarrow \triangle\Diamond_j E$ if $g(i) \supset g(j)$ $\triangle\nabla_i\nabla_j E \leftarrow \triangle\Box_k E$ if $g(k) = g(i) \cup g(j)$ $\triangle\nabla_i E \leftarrow \triangle\langle X\rangle_i\Box_i E$ if $g(i)$ is a singleton $\triangle\nabla_i\Diamond_i E \leftarrow \triangle\Diamond_i E$ if $g(i)$ is a singleton $\triangle\Diamond_i E \leftarrow \triangle\langle X\rangle_j\Diamond_i E$ if $g(i) \supseteq g(j)$

$L \in \{KD, T, KDB, B, KD4, S4\}$, L-MProlog	
Ext_L	$\triangle\Box E \rightarrow \triangle E$ if $L \in \{T, B, S4\}$ $\triangle\nabla\Box E \rightarrow \triangle E$ if $L \in \{KDB, B\}$ $\triangle\Box E \rightarrow \triangle\Box\Box E$ if $L \in \{KD4, S4\}$
Sat_L	the rules specifying Ext_L plus $\triangle E \rightarrow \triangle\Diamond E$ if $L \in \{T, B, S4\}$ $\triangle E \rightarrow \triangle\Box\Diamond E$ if $L \in \{KDB, B\}$ $\triangle\nabla\nabla' E \rightarrow \triangle\Diamond E$ if $L \in \{KD4, S4\}$
$rSat_L$	$\triangle\Diamond E \leftarrow \triangle\langle X\rangle E$ $\triangle E \leftarrow \triangle\Box E$ if $L \in \{T, B, S4\}$ $\triangle\Diamond E \leftarrow \triangle E$ if $L \in \{T, B, S4\}$ $\triangle E \leftarrow \triangle\langle X\rangle\Box E$ if $L \in \{KDB, B\}$ $\triangle\nabla\Diamond E \leftarrow \triangle E$ if $L \in \{KDB, B\}$ $\triangle\nabla\nabla' E \leftarrow \triangle\Box E$ if $L \in \{KD4, S4\}$ $\triangle\Diamond E \leftarrow \triangle\langle X\rangle\Diamond E$ if $L \in \{KD4, S4\}$

Table 5.3: Optimized sets of rules for L-MProlog, c.d.

for $L \in \{KD5, KD45, S5\}$ are also efficient enough. In the remainder of this section, we study optimizations for the remaining modal logics $KDI4_s$, $KDI4$, $KDI45$, $KD4I_g5_a$, KD, T, KDB, B, $KD4$, $S4$. Let L be one of these logics.

We adopt the following modifications for the original framework and its instantiations. In this way, we obtain the *optimized framework* and *optimized schemata* for L-MProlog.

1. When resolving a goal with an input clause, we relax the condition that "the mgu θ unifies the selected head atom A' with the forward labeled form A'' of the head of the input clause" by requiring only that θ is a most general substitution such that $A'\theta$ and $A''\theta$ have the same classical atom and $A'\theta$ is an L-instance of $A''\theta$.

2. In Tables 5.1, 5.2, and 5.3, we present optimized sets of rules for specifying operators Ext_L, Sat_L, and $rSat_L$. Atom variables X in the rules stand for fresh atom variables (i.e. standardizing atom variables apart is used). In general, an Ext_L/Sat_L rule of the form $\triangle\triangle'\alpha \rightarrow \triangle\triangle''\alpha$ in the original schema is replaced by $\triangle\triangle'E \rightarrow \triangle\triangle''E$, and an $rSat_L$ rule of the form $\triangle\triangle''\alpha \leftarrow \triangle\triangle'\alpha$ in the original schema is replaced by $\triangle\triangle''E \leftarrow \triangle\triangle'E$. Besides, we also discard the *general* $\Box$*-lifting rule* "$\triangle\nabla\alpha \leftarrow \triangle\Box_i\alpha$ if $\nabla \preceq_L \Box_i$" by embedding it into the other $rSat_L$ rules. This causes that Expected Lemmas 3.5.1 and 3.5.2 do not hold anymore and the proofs of Theorems 3.5.7 and 3.5.15 should be repaired.

3. We discard Expected Lemma 3.2.2 and Expected Theorem 3.2.3. We also discard the proof of Theorem 3.3.6 and rename this theorem to Expected Theorem 3.3.6.

4. We adopt the restriction that every resolution cycle is in the standard form.

5. For $L = KDI45$, we delete the accompanying condition of the NF_L rule $\triangle\nabla_i\nabla'_jE \rightarrow \triangle\nabla'_jE$ $(i \leq j)$ that ∇'_j is of the form $\Box_j$ or $\langle E\rangle_j$, because it does not affect the computation of $T_{L,P} \uparrow \omega$. We also discard the accompanying condition of the rNF_L rule $\triangle\nabla_jE \leftarrow \triangle\langle X\rangle_i\nabla_jE$ ($L = KDI45$ and $i \leq j$) that ∇_j is of the form $\Box_j$ or $\langle E\rangle_j$. With the restriction stated in the preceding list item, this does not affect the output (i.e. the last goal and the composition of the used mgu's) of any resolution cycle. We do similar modifications for the schema for semantics of $KD4I_g5_a$-MProlog. This causes that Expected Lemma 3.5.4 does not hold anymore.

Theorem 5.2.1 *For $L \in \{KDI4_s$, $KDI4$, $KDI45$, $KD4I_g5_a$, KD, T, KDB, B, KD4, S4$\}$, (Expected) Theorems 3.3.6, 3.5.7, 3.5.15, and 3.5.16 (on correctness of the fixpoint semantics, soundness, completeness, and strong completeness of SLD-resolution) hold when the optimized framework and the optimized schema for semantics of L-MProlog are used.*

Sketch. First, consider how to repair the proofs of Theorems 3.5.7, 3.5.15, and 3.5.16. Assume that the general $\Box$-lifting rule "$\triangle\nabla\alpha \leftarrow \triangle\Box_i\alpha$ if $\nabla \preceq_L \Box_i$" is still included in the set of rules for specifying $rSat_L$ and the rules specifying rNF_L were not changed. Then Expected Lemmas 3.5.1, 3.5.2, and 3.5.4 can easily be verified and the proofs of Theorems 3.5.7, 3.5.15, 3.5.16 still work. Since the left hand side of any $rSat_L/rNF_L$ rule does not explicitly contain any universal operator, any SLD-refutation in L can be simulated by another one with the same computed answer that does not use the general $\Box$-lifting rule. Hence, without the made assumptions, Theorems 3.5.7, 3.5.15, and 3.5.16 still hold, provided that Expected Lemmas 3.3.1, 3.3.2, 3.3.5, 3.5.3 can be proved.

It is clear that Expected Lemma 3.3.1 still holds.

The given proofs of Expected Lemma 3.3.2 still work.

Expected Lemma 3.5.3 can easily be verified.

The given proofs of Expected Lemma 3.3.5 need an adaptation because they use (discarded) Expected Lemma 3.2.2. Here, the given proofs of Expected Lemma 3.2.2 can be used partially. Consider, for example, the case $L \in \{KDI4_s, KDI4\}$. We change the last paragraph of the proof of Expected Lemma 3.3.5 given at page 59 by the following:

> *Since $M, w \vDash B_i$ for $1 \leq i \leq n$, it follows that $\triangle B_i$ is an L-instance of some atom from $Sat_L(I)$. Consequently, $\triangle A$ is an L-instance of some atom $\alpha \in T_{0L,P}(Sat_L(I)) = T_{L,P}(I) \subseteq I$. We show that $M, w \vDash A$. Let H be the function specifying contents of possible worlds in the standard L-model graph of I. We have that A is an L-instance of some atom from $H(w)$.*
>
> *(Here is a modified fragment of the proof of Expected Lemma 3.2.2 given at page 58.) The cases when A is a classical atom or $A = \Diamond_i E$ are trivial. Suppose that $A = \Box_i E$. Let u be a world such that $R_i(w, u)$ holds. It suffices to show that $E \in H(u)$. Since $R_i(w, u)$ holds, there are two cases:*
>
> - *Case $L = KDI4_s$, $w_0 = w$, $R'_{j_1}(w_0, w_1)$, $R'_{j_2}(w_1, w_2)$, ..., $R'_{j_h}(w_{h-1}, w_h)$, $R'_j(w_h, u)$, $h \geq 0$ and $j \leq i$: Since $\Box_i E \in H(w)$, by Lemma 3.2.1, there exists a $\Box$-lifting form $\triangle'$ of $\triangle$ such that $\triangle' \Box_i E \in Ext_L(I)$. By the definition of $Ext_L(I)$, it follows that $\triangle' \Box_{j_1} \ldots \Box_{j_h} \Box_j E \in Ext_L(I)$. Hence, by Lemma 3.2.1, $E \in H(u)$.*
> - *Case $L = KDI4$, $w_0 = w$, $R'_{j_1}(w_0, w_1)$, $R'_{j_2}(w_1, w_2)$, ..., $R'_{j_h}(w_{h-1}, w_h)$, $R'_j(w_h, u)$, $h \geq 0$ and $k \leq i$ for all $k \in \{j_1, \ldots, j_h, j\}$: The assertion holds by the same argumentation as for the above case.*

We can adapt the existing proofs of Expected Lemma 3.3.5 for the other logics analogously.

Now consider Expected Theorem 3.3.6, which states that $M_{L,P}$ is a least L-model of P. Recall that, for the original framework, its proof has the form *"By Lemma 3.3.5, $M_{L,P}$ is an L-model of P. Let M be an arbitrary L-model of P. By Lemma 3.3.4, $M \vDash I_{L,P}$. Hence, by Theorem 3.2.3, $M_{L,P} \leq M$. Therefore $M_{L,P}$ is a least L-model of P."*

The given proofs of Expected Lemma 3.3.4 still work.

If we change (discarded) Expected Theorem 3.2.3 to *"For P being an L-MProlog program, $M_{L,P}$ is a least L-model of $I_{L,P}$."*, then its proofs can be adapted as follows: M is replaced by $M_{L,P}$, I is replaced by $I_{L,P}$, and the sentence *"By Lemma 3.2.2, M is an L-model of I."* is replaced by *"By Lemmas 3.3.5 and 3.3.4, $M_{L,P}$ is an L-model of $I_{L,P}$."*.

Therefore, Expected Theorem 3.3.6 holds, using the existing proof. •

5.3 Semantics of MProlog-□

An L-MProlog-□ program (resp. goal) is an L-MProlog program (resp. goal) without existential modal operators. In this section, let L denote one of the modal logics considered in this work except $BSMM$. We show that the schemata given for semantics of L-MProlog can be significantly simplified for L-MProlog-□.

Let P denote an L-MProlog-□ program and G an L-MProlog-□ goal. Consider the (original or optimized) schema for semantics of L-MProlog. (For L being one of the basic serial monomodal logic, the schema is implicitly given in Section 4.5.) Observe that:

1. It can be proved by induction on n that $T_{L,P} \uparrow n$ contains only atoms of the form $\boxdot E$. Hence, the same claim can be said for $I_{L,P}$.

2. Possible worlds of $M_{L,P}$ are identified by modalities of the form $\langle\top\rangle_{i_1}\ldots\langle\top\rangle_{i_k}$ (which can be treated as $\Box_{i_1}\ldots\Box_{i_k}$).

3. For all of the rules specifying Sat_L, if the l.h.s. atom contains an unlabeled existential modal operator then the r.h.s. atom also contains an unlabeled existential modal operator. As no atoms with unlabeled existential modal operators can be used to compute $T_{0\,L,P}$, all Sat_L rules with unlabeled existential modal operators can be deleted without affecting the computation of $T_{L,P}{\uparrow}\omega$.

4. No labeled existential modal operators appear in the computation of $T_{L,P}{\uparrow}\omega$. Hence, in all of the rules specifying NF_L, the case with $\langle E\rangle_i$ (of the original schema) can be deleted. Furthermore, ∇_i, ∇'_i, and ∇'_j in the NF_L rules can be replaced respectively by $\Box_i$, $\Box_i$, and $\Box_j$; the NF_L rule $\nabla\nabla' E \rightarrow \nabla' E$ for $L \in \{KDI4_s5, KD4_s5_s\}$ can be replaced by $\Box_i\Box_j E \rightarrow \Box_j E$; for the case when $L \in \{KD5, KD45, S5\}$, both ∇ and ∇' in the rules specifying NF_L can be replaced by $\Box$. These modifications do not affect the computation of $T_{L,P}{\uparrow}\omega$, as both P and the rules specifying Sat_L contain no unlabeled existential modal operators.

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

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

7. With the modifications mentioned in the two preceeding items, every SLD-derivation from $P\cup\{G\}$ in L does not contain any (labeled or unlabeled) existential modal operator. Consequently, the modal operators ∇_i, ∇'_i, and ∇_j in the (remaining) rules specifying $rSat_L$ or rNF_L can be replaced respectively by $\Box_i$, $\Box_i$, and $\Box_j$. In general, the rules can be reformulated so that they contain no operators in the form ∇.

8. After the mentioned modifications, newly obtained rules of the form $\alpha \rightarrow \alpha$ or $\alpha \leftarrow \alpha$ should be deleted.

(Not all the mentioned observations, e.g. items 3 and 5 of the list, hold for the schema of semantics of $BSMM$-MProlog given in Table 4.1.)

The mentioned modifications create schemata for semantics of L-MProlog-$\Box$. Such a schema is much simpler than the corresponding schema for semantics of L-MProlog, as it does not involve (labeled or unlabeled) existential modal operators. More importantly, the resulting SLD-resolution calculus for L-MProlog-$\Box$ is much more efficient than the corresponding SLD-resolution calculus for L-MProlog. As an example, we present in Table 5.4 the resulting optimized schema for semantics of $KDI4_s$-MProlog-$\Box$.

5.4 Restrictions and Iterative Deepening Search

It is not easy for users to imagine and control the behaviour of MProlog programs. One of the reasons is that MProlog uses $rSat_L/rNF_L$ rules as meta clauses and the user may not fully

$L = KDI4_s$, L-MProlog-$\Box$
$\preceq_L$ is specified by $\Box_i \preceq_L \Box_j$ for $i \leq j$.
No restrictions on L-normal form of modalities.
No rules specifying NF_L and rNF_L.
Rules specifying
Ext_L and Sat_L
$\triangle\Box_i E \rightarrow \triangle\Box_j E$ if $i > j$
$\triangle\Box_i E \rightarrow \triangle\Box_j\Box_i E$
$rSat_L$ $\triangle\Box_j\Box_i E \leftarrow \triangle\Box_i E$

Table 5.4: An optimized schema for semantics of $KDI4_s$-MProlog-$\Box$

aware of all possible effects of the rules. Even when they understand the rules well, they may not have enough control on the rules without modifying the interpreter of the used MProlog system. Another reason is that a complete SLD-resolution for L-MProlog may need a rule like $\triangle\Box_i E \leftarrow \triangle\Box_i\Box_i E$, which can be applied repeatedly forever. This difficulty suggests that programming in MProlog is better treated as a mix of "programming" and theorem proving for the modal Horn fragment.

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

- a limit on the lengths of modalities that can occur in derivations,
- a limit on the number of applications of $rSat_L$ (resp. rNF_L) rules in a resolution cycle,
- a limit on the nesting depth of terms occurring in modal atoms,
- a limit on the length of derivation,
- a limit on the number of applications of $rSat_L/rNF_L$ rules in the derivation.

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

5.5 Tabulation

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

There are advanced tabulation methods for Prolog like OLDT-resolution [70], linear tabulated resolution [69, 73]. These methods use sophisticated techniques (e.g. the suspension-resumption mechanism and the stack-wise representation of OLDT) that are better implemented by the underlying Prolog abstract machine. Besides, these methods try to reach "answer completion" for subgoals as early as possible.

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

Example 5.5.1 Consider the goal $G = \leftarrow \Diamond s(x)$ and the following program P in the modal logic *KD*:

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

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

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

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

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

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

Chapter 6

Design and Implementation of the MProlog System

Starting from the purely logical formalism of MProlog, we have built a prototyping system for it [58]. The implemented system adds extra features to the purely logical formalism in order to increase usefulness of the language. It is written in Prolog and can run on SICStus Prolog and SWI-Prolog.[1] In this chapter, we use MProlog to refer to the implemented system.

6.1 The Design of MProlog

MProlog is designed as an extension of Prolog. This means that we can use Prolog codes, libraries and most features of Prolog in MProlog programs. This gives MProlog capabilities for real applications. MProlog is implemented as a module for Prolog and does not have its own running environment. It provides instead a list of built-in predicates to be used in Prolog.

6.1.1 Syntax of MProlog Programs

In MProlog, there are three kinds of predicates: classical predicates, modal predicates, and classical predicates which are defined using modal formulas. Predicates of the last kind are called *dum* predicates. The semantics of classical predicates and *dum* predicates does not depend on worlds in Kripke models. If E is a classical atom of a classical predicate or a *dum* predicate then $\triangle E \equiv E$ for every modality $\triangle$. An MProlog program consists of *modal fragments* and *classical fragments* (in an arbitrary number and order). Predicates defined in classical fragments are *classical predicates*. *Dum predicates* are declared in an MProlog program as follows

:- dum_pred $Pred_1$, ..., $Pred_n$.

where each $Pred_i$ is a pair $Name_i/Arity_i$. *Dum* predicates are defined in modal fragments by clauses of the form *E :- Body*, where E is a classical atom. A predicate defined in a modal fragment and not declared earlier as a *dum* predicate is a *modal predicate*. The same Name/Arity pair can be used for a classical predicate and a modal/dum predicate, but the classical predicate will not be accessible from modal fragments.

From now on, by a *calculus* we mean an SLD-resolution calculus for MProlog. An MProlog program may use different calculi (mixing different calculi using *dum* predicates is explained

[1] The last version 2.0 of MProlog has been tested using SICStus Prolog 4.0.2 and SWI-Prolog 5.6.52 on Windows XP.

in Section 6.2.2). In an MProlog program, a modal fragment starts with a declaration of the form:

:- calculus Cal_1, ..., Cal_n.

where Cal_1, ..., Cal_n are names of calculi. These calculi are called the *calculi of the fragment.* If an MProlog program is loaded by *mconsult(File,Cal)* then the program in *File* is treated as if it begins with

:- calculus Cal.

A modal fragment ends either by a declaration of another modal fragment, or by the end of the program, or by one of the two following declarations:

:- calculus classical.

:- end.

Programs may change values of options of the used calculi using:

:- set_option(Option, Calculus, Value).

In MProlog, modalities are represented as lists, e.g., as follows:

$\Box\Diamond q(x,y)$	$[b,d] : q(X,Y)$
$\Box_i\langle X\rangle_3\Diamond_j q(a)$	$[bel(I), pos(3,X), pos(J)] : q(a)$
$\Box_x god\_exists \leftarrow christian(x)$	$[bel(X)] : god\_exists$:- $christian(X)$

Here, b stands for "box", d for "diamond", *bel* for "believes", and *pos* for "possible". We use $\triangle : \varphi$ to represent $\triangle\varphi$. Notations of modal operators depend on how the base SLD-resolution calculus is defined. As another example, for MProlog-$\Box$ we represent $\Box_{i_1} \ldots \Box_{i_k}$ as $[I1, \ldots, Ik]$ (see *belief_box.cal* of [58]).

Syntactically, an MProlog program is a Prolog program. Modal fragments in an MProlog program may contain directives and clauses. Each clause in a modal fragment is of one of the following forms:

Context : (Head :- Body).

Head :- Body.

where *Context* is a list representing a modality, *Head* is of the form E or $M : E$, where E is a classical atom (in the sense of Prolog) and M is a list containing one modal operator. All clauses in a modal fragment are *declared* to all of the calculi of the fragment.

6.1.2 Syntax of SLD-Resolution Calculi for MProlog

SLD-resolution calculi for MProlog are specified using the framework given in Section 3 and written in Prolog. We have built such calculi for MProlog in the considered modal logics of belief and the basic serial monomodal logics (see *belief.cal*, *groups.cal*, *smnm.cal* of [58]). We have built also calculi for MProlog-$\Box$ in the considered modal logics of belief (see *belief_box.cal* of [58]). Users can define their own calculi. One file may contain some calculi, and due to the command *include*, such a file may include other files.

An SLD-resolution calculus for L contains $rSat_L/rNF_L$ rules, definitions of auxiliary predicates, and definitions for the following required predicates:

1. *universal_modal_operator(Calculus, Operator)*
2. *dual_modal_operator(Calculus, Operator, DualOperator)*
3. *box_lifting_form(Calculus, ModalOperator, BoxLiftingForm)*
   which returns true iff *BoxLiftingForm* is the universal modal operator of the same modal index as *ModalOperator* in the calculus *Calculus*

4. *forward_labeled_form(Calculus, SimpleModalAtom, ForwardLabeledForm)*
5. *normal_labeled_form(Calculus, Modality)*
6. *operator_instance(Calculus, OperatorInstance, ModalOperator)*
7. *context_instance(Calculus, ContextInstance, ModalContext)*

If the option *check_in_two_steps* of the defined calculus is set to *true*, then instead of the last three predicates of the above list, the calculus must implement two predicates with name prefixed by *pre_check_* or *post_check_* for each of the replaced predicates.

Let us discuss the usefulness of the *check_in_two_steps* option. Suppose that we want to reason about multi-degree belief. We represent, e.g., $\Box_i(p(x) \leftarrow q(x))$ by [bel(I)]: (p(X) :- q(X)). Thus we use variables like I for degrees of belief. Sometimes it is better to delay instantiating such variables to concrete values in order to eliminate branching. If we want to allow users to have the ability to turn on/off this option of delaying, then the defined calculus should be designed with the *check_in_two_steps* option turned on. The intention of *pre_check* predicates is to check the involved condition as much as possible without generating branching points and to pass unchecked fragments of the condition to *post_check* predicates, which will be fired latter.

To create an ability to reduce nondeterminism, we provide 4 categories (kinds) of rules: *pre_rSat*, *rSat*, *post_rSat*, and *rNF*. Informally, operators *pre_rSat* and *post_rSat* are deterministic, while operators *rSat* and *rNF* are nondeterministic. This means that when the system tries to resolve a modal atom using an operator of the category *pre_rSat* or *post_rSat*, the first applicable rule of the category will be used, and when the system wants to use *rSat* (resp. *rNF*), different sequences of *rSat* rules (resp. *rNF* rules) will be tried. Lengths of such sequences of *rSat* rules (resp. *rNF* rules) are restricted by the option *limit_rSat* (resp. *limit_rNF*) of the calculus.

Rules of the mentioned categories are of one of the following forms:

AtomIn :- PreCondition, AtomOut, PostComputation.

RuleName :: (AtomIn :- PreCondition, AtomOut, PostComputation).

AtomIn and *AtomOut* are atoms of the form $M : E$, where M (standing for a modality) and E (standing for a classical atom) may be variables in Prolog, and M may be also a list. *RuleName* is a name in Prolog. *PreCondition* and *PostComputation* are (possibly empty) sequences of formulas in Prolog separated by ','. *AtomOut* is called the *atom out* of the rule. It is the first outer (w.r.t. ',') atom of the form $M : E$ of the body of the rule. Names of rules are unique. If a rule is declared without a name, it will be given a unique name by the system.

We give below a definition for the rule ($\triangle\nabla_i\nabla'_i E \leftarrow \triangle\nabla_i E$ if ∇'_i is of the form $\Box_i$ or $\Diamond_i$) of the SLD-resolution calculus of $KD45_{(m)}$-MProlog:

```
M:E :-
    append(M2, [O1,O2], M),
    (O1 = bel(I); O1 = pos(I); O1 = pos(I,_)),
    (O2 = bel(I); O2 = pos(I)),
    append(M2, [O2], M3),
    M3:E.
```

Here is another example for a version of the rule ($\Diamond_i E \leftarrow \Diamond_j E$ if $i > j$) :

```
rSatKDI4s5 :: ( [pos(I)]:E :-
    get_calling_history(rSat, Cal, _, RNames),
```

```
\+ memberchk(rSatKDl4s5, RNames), % not called before
pre_compare_deg(Cal, I > J), [pos(J)]:E, post_compare_deg(Cal, I > J)).
```

As shown in the above example, designers of calculi have access to the history of rules called in the current resolution cycle. Such a history for a given category of rules is obtained by

get_calling_history(RuleCategory, Calculus, CalledAtom, RuleNames)

where the last three arguments are outputs. *RuleNames* is the list of names of rules of the category *RuleCategory* which have been applied, in the reverse order, in the current resolution cycle for the beginning atom *CalledAtom*. The mentioned predicate *get_calling_history*/4 requires the boolean option *use_calling_history* of the calculus to be turned on.

Syntactically, rules are clauses in Prolog. They are *defined* as usual clauses. Rules are *declared* to calculi either by a *section of rules* or by a directive. A section of rules is a list of (definitions of) rules bounded by directives. A directive opening a section of rules is of the following form:

:- RuleCategory Cal_1, ..., Cal_n.

where *RuleCategory* is one of *pre_rSat*, *rSat*, *post_rSat*, *rNF*; and Cal_1, ..., Cal_n are names of calculi, to which the rules in the section are declared. A section of rules is closed by any directive in Prolog or by the end of the main file. Rules of the same category can also be declared to a calculus using a directive of the following form:

:- set_list_of_mrules(Calculus, RuleCategory, ListOfRuleNames).

Some options are automatically created with default values for each loaded calculus. A definition of a calculus can change values of those options using *set_option(Option, Calculus, Value)* and set new options for itself (e.g., a numeric option like *max_modal_index* is needed for modal logics of multi-degree belief).

6.1.3 Main Predicates of MProlog

There are three groups of built-in predicates which are useful for users: main predicates (for consulting, calling, and tracing), predicates for getting and displaying the status of the system, and predicates for modifying the program. We leave the two latter groups to the next section, where we discuss data structures of MProlog, and list here only main predicates of the system:

consult_calculi(Files),
mconsult(ProgramFile, Calculus), *mconsult(ProgramFile)*,
mcall(Goal, Calculus), *mcall(Goal)*,
mtrace, *nomtrace*.

Our MProlog module can be loaded by consulting the file "mprolog.pl" of the package. The user can then load SLD-resolution calculi for modal logics using the predicate *consult_calculi*, whose argument may be a file name or a list of file names. The user can consult MProlog programs using the predicate *mconsult*/2 (see Section 6.1.1 for the meaning of the second argument). *mconsult(ProgramFile)* is treated as *mconsult(ProgramFile, classical)*. Goals involving with modal logics can be asked using the predicate *mcall*/2, where the second argument indicates the calculus in which the goal is asked. If a default calculus is set using *set_option(current_calculus, Calculus)*, then *mcall*/1 can be used instead of *mcall*/2. The predicates *mtrace* and *nomtrace* are used to turn on and off the trace mode for MProlog (which concentrates on modal formulas and is not the trace mode of Prolog).

For iterative deepening search w.r.t. the number of applications of *rSat*/*rNF* rules in the derivation, *mcall2* is used instead of *mcall*. Similarly, for iterative deepening search w.r.t. the length of derivation, *mcall3* is used instead.

The tabulation mechanism may affect the set of results when the program is modified. To refresh the tabulation mechanism, the user can call *delete_tabulation*.

6.1.4 Options of MProlog

Behaviors of the MProlog interpreter are controlled by options. There are two kinds of options: options for calculi, and options for the system. Setting and getting values of options are done by the following predicates. The first two ones work for options of calculi, while the last two ones work for options of the system:

set_option(OptionName, Calculus, Value),
get_option(OptionName, Calculus, Value),
set_option(OptionName, Value),
get_option(OptionName, Value).

There are the following built-in options for calculi:

limit_modality_length	default value:	4
limit_rSat		3
limit_rNF		1
use_calling_history		false
check_in_two_steps		false

The *limit_modality_length* option is used to restrict lengths of modalities appearing in derivations. The options *limit_rSat* and *limit_rNF* have been described in Section 6.1.2. For some built-in calculi, these numeric limits are firmly set, as they follow from the nature of the base modal logic. In general, they are used to restrict the search space and may affect completeness of the calculus. The boolean option *use_calling_history* should be turned on if rules of the calculus use the history of rules called in the current resolution cycle. The boolean option *check_in_two_steps* has been discussed in Section 6.1.2.

The version 2.0 of MProlog has the following system options:

limit_term_depth	default value:	5
limit_rule_applications		1000000
limit_proof_size		1000000
deepening_step_for_rule_applications		30
deepening_step_for_proof_size		50
random_selection_of_rules		false
sorting_mclauses		false
priority_list_of_calculi		
current_calculus		classical
debug		false
quiet		true
show_answer_duplicates		false

The *limit_term_depth* option establishes a limit on the nesting depth of terms occurring in modal atoms. The *limit_rule_applications* establishes a limit on the number of applications of *rSat*/*rNF* rules in the derivation. The *limit_proof_size* establishes a limit on the length of derivation (not counting executions of classical fragments). These options are used to restrict the search space and may affect completeness of the calculus.

The option *deepening_step_for_rule_applications* (resp. *deepening_step_for_proof_size*) is used for iterative deepening search w.r.t. the number of applications of *rSat*/*rNF* rules in the

derivation (resp. the length of derivation).

When selecting a rule of the category *rSat* or *rNF*, if the boolean option *random_selection_of_rules* is off, then rules will be considered accordingly to their declaration order, otherwise they will be randomly considered.[2] When an MProlog program is "consulted", if the boolean option *sorting_mclauses* is turned on, then modal clauses will be sorted according to the lengths of their bodies. (This allows unary modal clauses to be considered first.) The option *priority_list_of_calculi* will be discussed later.

6.1.5 Examples of MProlog Programs

The Three Wise Men Puzzle

We give here an MProlog program that formalizes the three wise men puzzle using the multi-modal logic $KD4I_g5_a$ of belief and common belief. An SLD-resolution calculus for this logic has been developed and is denoted by cKD4Ig5a [58]. In this calculus, *bel* denotes belief and *pos* denotes possibility. The calculus requires definitions of predicates *singleton_group*/1, *subgroup*/2, and *union_group*/3. Denote the wise men by a, b, c, and the possible groups by gAB, gAC, gBC, $gABC$, where, e.g., $gABC = \{a, b, c\}$. Thus, $[bel(gABC)] : \varphi$ means that φ is a common belief of the group $\{a, b, c\}$. Define the mentioned required predicates in the usual way. The three wise men problem can be formalized by the following MProlog program:

```
:- calculus cKD4Ig5a.

% If Y sits behinds X then X's card is white if Y considers this as possible.
[bel(gABC)]: (white(X) :-
        member(X, [a,b,c]), member(Y, [a,b,c]), X @< Y, [pos(Y)]:white(X)).

% The following formula is "dual" to the above formula.
[bel(gABC)]: ([bel(Y)]:black(X) :-
        member(X, [a,b,c]), member(Y, [a,b,c]), X @< Y, black(X)).

% At least one of the wise men has a white card.
[bel(gABC)]: (white(a) :- black(b), black(c)).
[bel(gABC)]: (white(b) :- black(c), black(a)).
[bel(gABC)]: (white(c) :- black(a), black(b)).

/* Each of B and C does not know the color of his own card. In particular, each of the men
considers that it is possible that his own card is black. */
[bel(gABC),pos(c)]:black(c).
[bel(gABC),pos(b)]:black(b).

% Change some default options:
:- set_option(limit_modality_length, cKD4Ig5a, 3).
:- set_option(limit_rSat, cKD4Ig5a, 1).
:- set_option(current_calculus, cKD4Ig5a).
```

The question is whether A believes that his card is white. It is passed to the Prolog interpreter as $mcall([bel(a)] : white(a))$. The MProlog system spends less than 1 second for resolving this goal, using SICStus Prolog on TravelMate 230X, 1.7GHz-M.

[2] but with a slight emphasis of the declaration order

The n Wise Men Puzzle

The n wise men puzzle is the same as the version of 3 wise men, except that there are n wise men. Denote the wise men by natural numbers from 1 to n. A group of wise men is denoted by the ordered list consisting of the identifications of the men. A singleton group $[X]$ can be treated as wise man X. Here is the whole program in MProlog for the n wise men puzzle:

```
% Here are predicates required for the calculus cKD4Ig5a:
singleton_group([_]).
subgroup(G1, G2) :- subList(G1, G2), G1 \= G2, G1 = [_ | _].
subList([], _).
subList([X], L) :- member(X, L).
subList([X, Y | L], L2) :- member(X, L2), member(Y, L2), X < Y, subList([Y | L], L2).
union_group(G1, G2, G3) :- merge(G1, G2, [], G3).
merge([X | L1], [Y | L2], Acc, L) :- X = Y, !, merge(L1, [Y | L2], Acc, L).
merge([X | L1], [Y | L2], Acc, L) :- X < Y, !, merge(L1, [Y | L2], [X | Acc], L).
merge([X | L1], [Y | L2], Acc, L) :- merge([X | L1], L2, [Y | Acc], L).
merge(L1, [], Acc, L) :- reverse(Acc, A), append(A, L1, L).
merge([], L2, Acc, L) :- reverse(Acc, A), append(A, L2, L).
% Here are auxiliary predicates to fix n and set up the group of all wise men.
% We also set the limit_modality_length option to n.
set_number_of_men(N) :-
        generate_men(N, All),
        set_info(all_wise_men, All),
        set_option(limit_modality_length, cKD4Ig5a, N).
generate_men(N, All) :- generate_men(N, [], All).
generate_men(0, Acc, Acc) :- !.
generate_men(K, Acc, All) :- K2 is K - 1, generate_men(K2, [K | Acc], All).
/* Here are main predicates of the program. We use the predicate white(ListOfMen) to denote
that at least one of the men in the list ListOfMen has a white card, and use black(X) to denote
that wise man X has a black card. */
:- calculus cKD4Ig5a.
% At least one of the men has a white card.
[bel(All)]: (white(All) :- get_info(all_wise_men, All)).
/* Let G be a group of men, X be a man in G, and G2 = G - {X}. If at least one of the
members of G has a white card and the card of X is black, then at least one of the members
of G2 has a white card. */
[bel(All)]: (white(G2) :-
        get_info(all_wise_men, All), subList(G2, All), black(X),
        subList(G, All), member(X, G), delete(G, X, G2), white(G)).
% If Y sits behinds all members of the group G, then white(G) holds if Y considers that as
possible.
[bel(All)]: (white(G) :-
        get_info(all_wise_men, All), subList(G, All), append(_ , [X], G),
        member(Y, All), X < Y, [pos([Y])]:white(G)).
```

```
/* Each of the wise men with a number different than 1 does not know the color of his own
card. In particular, each of those men considers that it is possible that his own card is black.
*/
[bel(All)]: ([pos([X])]:black(X) :- get_info(all_wise_men, All), member(X, All), X \= 1).

% Change some default options:
:- set_option(limit_rSat, cKD4lg5a, 1).
:- set_option(current_calculus, cKD4lg5a).
```

The question is still whether wise man number 1 believes that his card is white. Here is an example of goal with $n = 10$:

```
?- set_number_of_men(10), mcall([bel([1])]:white([1])).
```

Here are test results for this program on Acer TravelMate 2482WXMi, 1.73 GHz, using SICStus Prolog 4.0.2:

- for the uncompiled program:
  $n = 10$: 1s; $n = 11$: 2s; $n = 12$: 4s; $n = 13$: 8s; $n = 14$: 16s;
- for the compiled program, without inclusion of the trace module: $n = 14$: 4s.

It can be seen that the compiled version runs much more faster than the uncompiled version, and the complexity of the program is exponential.

6.2 Implementation of MProlog

In this section, we describe our implementation of the MProlog system. The system consists of the following modules: data structures, the parser, the interpreter, the interface module, the tracer, and the compiler. We describe the interpreter in details to clarify the functioning of the system. For the other modules, we give only brief descriptions. Details on the predicates implemented in those modules are given in Appendix A.

6.2.1 Data Structures and the Parser of MProlog

The implementation of MProlog uses certain data structures. There are four main data entities used for MProlog: calculi, predicates, rules, and modal clauses. We divide data representation into two levels: the physical level and the logical level. The physical level implements details of data representation and communicates only with the logical level. The logical level exports necessary predicates for data manipulation to other modules of the system, while hiding details of data representation. The logical level plays the role of a communication protocol between the module of data structures and the other modules of the systems. This approach makes the system more modular, as we can optimize data representation without modifying the other modules.

Predicates for data manipulation can be divided into two groups: for setting and for getting data. In the specific case of MProlog, predicates for getting data are used for the interpreter and as user-end predicates for getting and displaying status of the system. Predicates for setting data of MProlog are used for the parser and as user-end predicates for dynamic modification of programs.

The module of data structures of MProlog contains also predicates for tabulation and for manipulating history of rules called in the current resolution cycle. Those predicates are used mainly in the MProlog interpreter.

The parser of the MProlog system is used to parse MProlog programs and calculi. As mentioned earlier, MProlog programs and calculi have Prolog syntax. To parse them we use *peek_code*/2 and *get_code*/2 to omit comments and use *read*/2 to read terms, which are delimited by '.'. Each term read from a parsed file is either a directive or a clause.

Directives are recognized as terms of the functor ':-'/1. The type of a directive is specified by the name appearing next to ':-'. In general, after omitting comments and separators like spaces, if the next character is ':' then we can assume that the next element will be a directive. This means that we can check whether the next element is a directive or a clause by 'peeking' only.

A clause read from a parsed file can be one of the following: a rule not in a section of rules, a rule in a section of rules, a modal clause (in a modal fragment), or a classical clause (in a classical fragment of a program or in a file of calculi for defining required or auxiliary predicates of the calculi). The first case can be recognized without context, because a rule not defined in a section of rules must be named and is a term of the functor '::'. The last three cases can be distinguished by the context.

For each element read from a parsed file, depending on its type and the context, appropriate actions will be executed (for example, to declare a predicate or to store a rule or a modal clause).

6.2.2 The Interpreter of MProlog

For clarity, we describe the version 1.0 but not the current version 2.0 of the MProlog interpreter. (The 2.0 of the MProlog interpreter extends the version 1.0 with tabulation for modal atoms and the ability of iterative deepening depth-first search.)

The MProlog interpreter is realized by the predicate *mcall(Goal, Calculus)*, which initiates some variables and then calls *mcall_ (Goal, Calculus)*. In this subsection, we describe in detail the latter predicate, ignoring some aspects like tabulation, updating the history of called rules, or effects of options.

The predicate of the argument *Goal* belongs to one of the following groups: control predicates[3] (\+, ';', ',', =>), classical predicates, *dum* predicates, and modal predicates.

If *Goal* is an atom of a classical predicate, then *mcall_ (Goal,_)* is defined as *Goal* itself. The formulas *PreCondition* and *PostComputation* of a rule

Head :- PreCondition, AtomOut, PostComputation

are also treated as atoms of classical predicates, despite that they may be composite formulas.

Because *dum* predicates can be defined in different calculi and their semantics does not depend on worlds in Kripke models, they can be used to mix different calculi. If *Goal* is an atom of a *dum* predicate then to resolve *mcall_ (Goal, Calculus)*, the interpreter will try to resolve *Goal* first in *Calculus* and then in different calculi as well. The list of those latter calculi is determined by the value of the *priority_list_of_calculi* option if it is set, and by the list of all calculi otherwise; both cases exclude *Calculus* (the argument). Resolving *Goal* of a *dum* predicate in a calculus *Cal* is done as follows: select a modal clause *Head :- Body* of *Cal*, unify *Goal* with *Head*, and then call *mcall_ (Body, Cal)*.

For the case of modal atoms, we first discuss some auxiliary predicates.

Resolving a modal atom *Goal* with a rule

Head :- PreCondition, AtomOut, PostComputation

is done by unifying *Goal* with *Head*, executing *PreCondition*, and returning *AtomOut* and *PostComputation* as outputs. This task is done by the predicate

[3]From now on, we distinguish control predicates from classical predicates.

solve_using_mrule(Cal, Cat, RName, AtomIn, AtomOut, PostComputation)
with *AtomIn* = *Goal* and *Cal*, *Cat*, *RName* being respectively the calculus, the category, and the name of the rule.

Resolving a modal atom *Goal* using a sequence of rules is done by calling the above described predicate *solve_using_mrule* for each rule of the sequence, where *AtomIn* of the first call of *solve_using_mrule* is *Goal*, and *AtomIn* of each one of the next calls is *AtomOut* of the previous call. As outputs, it returns *AtomOut* of the last call of *solve_using_mrule* and the composition (using ',' and the reverse order) of the obtained *PostComputation* formulas. If the sequence of rules is empty then the outputs are *Goal* and *true*.

To resolve a modal atom *Goal* using rules of a calculus *Cal* that belong to a category *Cat*, the interpreter searches for a sequence of rules to be used using the following strategy: if the rule category is *pre_rSat* or *post_rSat* then the sequence consists of only the first applicable rule – if there exists, or is empty – otherwise; if the rule category is *rSat* or *rNF* then different sequences of rules will be tried, where a sequence S being a prefix of a sequence S' will be tried before S'. The interpreter applies a sequence of rules "in the fly". The task is done by the predicate

solve_using_mrules(Cal, Cat, Goal, AtomOut, PostComputation),

where *AtomOut* and *PostComputation* are outputs.

Resolving a modal atom *Goal* with a modal clause in *Calculus* is done according to the framework of SLD-resolution for MProlog, using the required predicates of *Calculus* in an appropriate way. The task is done by the predicate

solve_using_mclauses(Calculus, Goal).

Now return to the problem of resolving *mcall_ (Goal, Calculus)* for the case when *Goal* is a modal atom. It is done by executing the following statements

solve_using_mrules(Calculus, pre_rSat, Goal, A2, F2),
solve_using_mrules(Calculus, rSat, A2, A3, F3),
solve_using_mrules(Calculus, post_rSat, A3, A4, F4),
solve_using_mrules(Calculus, rNF, A4, A5, F5),
solve_using_mclauses(Calculus, A5),
F5, F4, F3, F2.

We now discuss the interpretation of the control predicates. In the current version of MProlog, we just adopt the following solution, which does not have a logical basis:

```
mcall_(M:(F1,F2), Cal) :- !, mcall_(M:F1, Cal), mcall_(M:F2, Cal).
mcall_(M:(F1;F2), Cal) :- !, mcall_(M:F1, Cal); mcall_(M:F2, Cal).
mcall_(M:(\+ F), Cal) :- !,
     make_dual_modality(Cal, M, M2), \+ mcall_(M2:F, Cal).
mcall_(M:(F1 -> F2), Cal) :- mcall_(M:F1, Cal) -> mcall_(M:F2, Cal).
```

The interpretation for the case of $M : (F1, F2)$ is sound and complete if M is a modality in labeled form (i.e. M does not contain unlabeled existential modal operators). The interpretation for the case of $M : (F1; F2)$ is also sound.

The tabulation mechanism of MProlog 2.0 is implemented as follows:

- When the interpreter has a new refutation for a modal atom, the resulting modal atom is tabulated.

- When the interpreter tries to solve a modal atom, if the atom or its variant was either called ealier in the current derivation or *locally completed*, then the atom is only resolved

with the tabulated modal atoms (that is, rules and modal clauses are not used for resolving the atom).

- When the first call in the current derivation of a modal atom has been totally resolved, (the process for solving) the atom is locally completed.
- At the highest level of calling *mcall_*, when backtracking occurs and the number of tabulated modal atoms has been increased, the interpreter deletes all information about "local completions" and re-does the call.

6.2.3 Other Modules of MProlog

Apart from the core modules (data structures, the parser, and the interpreter), MProlog contains also the interface module, the tracer, and the compiler, among which the two latter ones are not obligatory for running MProlog.

The interface module is built upon the three core modules of MProlog. It contains definitions of the main predicates listed in Section 6.1.3 and other friendly predicates. Some predicates defined in the module of data structures can be used also as user-end predicates for getting and displaying status of the system.

We omit describing the tracer because its current version requires essential improvements. We adopt a principle that the tracer should be modular, i.e. its implementation should not require modifications of the interpreter.

In Prolog, compiled programs run faster than their uncompiled versions. That is why we created a module for compiling MProlog calculi and programs. The main predicate defined in the module is *mcompile*/1, whose parameter is either *calculi* or *program*. The main task of this predicate is to dump the content of memory using *listing*/0 to a temporary file and compile that file. Because *listing* does not list declarations of used libraries, dynamic predicates, and operators, such declarations in the form of directives are written to the temporary file before calling *listing*.

Data structures used for MProlog are represented by dynamic predicates, which are processed more slowly than normal predicates. Some data structures used for MProlog need to be dynamic only for loading MProlog calculi and programs. This means that when compiling calculi or an MProlog program using *mcompile*/1, some predicates do not need to be dynamic anymore. The number of such predicates depends on what we want to compile – calculi or an MProlog program. This explains the parameter of the predicate *mcompile* described above.

When parsing calculi or an MProlog program, we use *assertz* to add classical clauses to the memory. This causes that all classical predicates defined in MProlog calculi and programs are implicitly declared as dynamic. The predicate *mcompile* overcomes this problem by consulting such classical predicates instead of asserting them.

Chapter 7

MDatalog and Modal Deductive Databases

In this chapter, we study modal deductive databases and a modal query language MDatalog, which is a sub-language of MProlog. We address subjects like data complexity of MDatalog, modal relational algebras, and evaluation methods for MDatalog.

In this chapter, L denotes one of the modal logics considered in this work except $BSMM$. We divide the group of such logics into two subgroups $\mathcal{BMD}$ (bounded modal depth) and $\mathcal{UMD}$ (unbounded modal depth) as follows:

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

The first group consists of KD, T, KDB, B and extensions of $KD5$. The second group consists of $sCFG$ and extensions of $KD4$.

7.1 Preliminaries

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 7.1.1 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 7.1.2 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 it as the atom $\triangle p(\mathfrak{t})$. Conversely, an ground atom $\triangle p(\mathfrak{t})$ 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.[1] 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 7.1.3 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 2.9.2 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 7.1.1 *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 3.3.6. •

Definition 7.1.4 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 $P \cup P_I \models_L \varphi(c_1, \ldots, c_k)$.

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

Proposition 7.1.2 *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 $c_1, \ldots, c_k$,*

$$P \cup P_I \models_L \varphi(c_1, \ldots, c_k) \text{ iff } P' \cup P_I \models_L q(c_1, \ldots, c_k).$$

This proposition can be proved in a similar way as Proposition 2.8.1 on the expressiveness of MProlog.

7.2 Data Complexity

Data complexity is measured when the query is fixed and the extensional database is taken as input. In [62], we showed that the data complexity of L-MDatalog for $L \in \{KD, T, KB, KDB, B, K5, KD5, K45, KD45, KB5, S5\}$ is complete in PTIME, for $L = K$ is complete in coNP, and for $L \in \{K4, KD4, S4\}$ is complete in PSPACE. In this section, we show that for $L \in \{KDI4_s5, KDI45, KD4_s5_s, KD45_{(m)}\}$ 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 $P \cup P_I \models_L \varphi(c_1, \ldots, c_k)$ 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 $L \in \{KDI4_s5, KDI45, KD4_s5_s, KD45_{(m)}\}$. Since the data complexity of Datalog is complete in PTIME (see, e.g., [38]), the data complexity of L-MDatalog for $L \in \{KDI4_s5, KDI45, KD4_s5_s, KD45_{(m)}\}$ is also complete in PTIME.

Theorem 7.2.1 *The data complexity of L-MDatalog for $L \in \mathcal{BMD}$ is in PTIME.*

Proof. The case $L \notin \{KDI4_s5, KDI45, KD4_s5_s, KD45_{(m)}\}$ was proved in [62]. Here, we assume that $L \in \{KDI4_s5, KDI45, KD4_s5_s, KD45_{(m)}\}$.

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 7.1.1, M is a least L-model of $P \cup P_I$. Hence, for every constant symbols $c_1, \ldots, c_k$, $P \cup P_I \models_L \varphi(c_1, \ldots, c_k)$ iff $M \models \varphi(c_1, \ldots, c_k)$. Checking $M \models \varphi(c_1, \ldots, c_k)$ 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 [62] 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. It can be shown that the data complexity of L-MDatalog for $L \in \{KDI4_s, KDI4\}$ is complete in PSPACE.

7.3 Modal Relational Algebras

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 7.3.2).

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 $1 \leq i \leq m$.[2] 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 $1 \leq i \leq m$.[3] 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, $Label_p(I) = \{(\triangle, \mathfrak{t}) \mid (\triangle, \mathfrak{t}) \in I$ and $\triangle$ 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.

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.

[2] In [59], $\Box_i^-$ and $\Diamond_i^-$ are denoted by $\Box_i$ and $\Diamond_i$, respectively.
[3] In [59], $\Box_i^+$ and $\Diamond_i^+$ are denoted by $\Box_i^{\leftarrow}$ and $\Diamond_i^{\leftarrow}$, respectively.

Definition 7.3.1 *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 7.3.2 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 7.3.1 *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 7.1.2, 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 $RE_L(I) = \{(\triangle, \mathfrak{t}) \in I \mid$ there is no $(\triangle', \mathfrak{t}) \in I$ 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.

7.4 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 7.4.1 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 7.4.2 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 7.4.1 *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 7.2.1.

By Proposition 7.1.2, every L-MDatalog query $(P, \varphi(x_1, \ldots, x_k))$ over a schema S can be effectively transformed 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 $c_1, \ldots, c_k$, $P \cup P_I \models_L \varphi(c_1, \ldots, c_k)$ iff $P' \cup P_I \models_L q(c_1, \ldots, c_k)$. Thus checking whether $P \cup P_I \models_L \varphi(c_1, \ldots, c_k)$ can be done by checking whether $q(c_1, \ldots, c_k)$ is true in the standard L-model of $P'_L(I) \cup I$ (by Lemma 7.1.1), which in turn is equivalent to $q(c_1, \ldots, c_k) \in P'_L(I)$. As $l\text{-}P'_L(I)$ is an approximation of $P'_L(I)$, *we can approximate the task of checking whether $P \cup P_I \models_L \varphi(c_1, \ldots, c_k)$ by checking whether $q(c_1, \ldots, c_k) \in l\text{-}P'_L(I)$, which is solvable in polynomial time in the size of the input I* (assuming that P and l are fixed).

Definition 7.4.3 Suppose that we are given an L-MDatalog query $(P, q(x_1, \ldots, x_k))$ 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 $P \cup P_I \models_L q(c_1, \ldots, c_k)$ by the check $q(c_1, \ldots, c_k) \in l\text{-}P_L(I)$ the *l-modal-depth-restricted fixpoint semantics* of the query. Under this semantics, the query $(P, q(x_1, \ldots, x_k))$ on the input I returns the set of tuples of constant symbols $(c_1, \ldots, c_k)$ such that $q(c_1, \ldots, c_k) \in l\text{-}P_L(I)$.

Definition 7.4.4 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 7.3.1 and can be proved analogously.

Proposition 7.4.2 *Every L-MDatalog query $(P, q(x_1, \ldots, x_k))$, 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.*

7.5 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 $L \in \mathcal{UMD}$.

7.6 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.

7.6.1 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 7.6.1 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 7.6.1 *Let P be an L-MDatalog program over a schema S with $\mathrm{edb}(S) = \emptyset$, 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 3.5.6, 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 $Sat_L(I_{L,P})$. •

7.6.2 A Formal Presentation of the Algorithm

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

Algorithm 7.6.1

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 7.6.2 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 $\{\mathfrak{t} \mid (\mathbf{.}, \mathfrak{t}) \in ans\_q$ where $\mathbf{.}$ is the empty modality$\}$.

Procedure 7.6.2

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 7.6.3 to process the goal $\leftarrow \alpha'$ on φ.

Procedure 7.6.3

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) $sup_0 := \{\delta_0\}$.[4]
     (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 7.6.2 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.

[4] sup_i denotes ith "supplementary" relation.

7.6.3 Properties of the Algorithm

In this subsection, we prove that the top-down evaluation method for MDatalog presented by Algorithm 7.6.1 is sound, complete, and tight. Roughly speaking, Algorithm 7.6.1 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 7.6.2 (Soundness) *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 execution of Algorithm 7.6.1 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 7.6.3. 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 7.6.3).

We can simulate the refutation of $\leftarrow \alpha$ embedded in Procedure 7.6.3 for $\leftarrow \alpha''$ as follows. Instead of the goal $\leftarrow (\triangle' B_1, \ldots, \triangle' B_n)\delta_0$ (or the sequence of goals $\leftarrow (\triangle' B_1)\delta_0$, $\ldots$, $\leftarrow (\triangle' B_n)\delta_{n-1}$), we have the ground goal $\leftarrow \triangle'' B_1\delta_n, \ldots, \triangle'' B_n\delta_n$.

Consider Step 2(b)iii of Procedure 7.6.3. We have that $\triangle'' B_i\delta_n$ is an L-instance of $\beta \in Sat_L(I(p_i))$. By Corollary 3.5.12, $P_I \cup \{\leftarrow \beta\}$ 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 7.6.3. 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 $P \cup P_I \cup \{\leftarrow \triangle'' B_1\delta_n, \ldots, \triangle'' B_n\delta_n\}$ 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 $\leftarrow \triangle'' B_i\delta_n, \ldots, \triangle'' B_n\delta_n$ to $\leftarrow \triangle'' B_{i+1}\delta_n, \ldots, \triangle'' B_n\delta_n$.) •

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 7.6.4.

Lemma 7.6.3 *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 end moment of an execution of Algorithm 7.6.1 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 $\theta_1, \ldots, \theta_h$ be the sequence of mgu's used in the refutation. We have that $(\triangle p(\mathfrak{t}))\theta_1 \ldots \theta_h = (\triangle p(\mathfrak{t}))\theta$. 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 $\theta_1, \ldots, \theta_j$, resulting in a goal $\alpha' = \Delta' A'$, and then uses a variant $\varphi' = \boxdot(A'' \leftarrow B'_1, \ldots, B'_n)$ of a program clause $\varphi = \boxdot(A \leftarrow B_1, \ldots, B_n)$ of P with mgu θ_{j+1}, resulting in the goal $\leftarrow (\triangle' B'_1, \ldots, \triangle' B'_n)\theta_{j+1}$. 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 $\leftarrow (\triangle' B'_i)\theta_{j_1-1}\theta_{j_1}\ldots\theta_{j_i-1}$ of the refutation of $\leftarrow \alpha$ uses mgu's $\theta_{j_i},\ldots,\theta_{j_{i+1}-1}$. Thus, after processing the atom B'_{i-1}, for $2 \leq i \leq n+1$, the next goal of the refutation of $\leftarrow \alpha$ is $\leftarrow (\triangle' B'_i,\ldots,\triangle' B'_n)\theta_{j_1-1}\theta_{j_1}\ldots\theta_{j_i-1}$.

Consider the last iteration of the main loop of Algorithm 7.6.1, the execution of Procedure 7.6.2 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 7.6.3 (in that execution of Procedure 7.6.2).

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 7.6.3, 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:

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

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 3.5.9, 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 7.6.1, $(\triangle' B_i)\delta_{i-1}\theta'_{j_i}\ldots\theta'_{j_{i+1}-1}$ 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:

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

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 3.5.9, 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 7.6.2 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 7.6.3 is a $\square$-lifting form of $\alpha\gamma_0\delta_n$ and is ground, so it is also a $\square$-lifting form of $\alpha\gamma_0\delta_n\gamma'_n = \alpha\gamma_0\varrho^{-1}\delta_n\gamma'_n = (\triangle p(\mathfrak{t}))\theta_1\ldots\theta_h = (\triangle p(\mathfrak{t}))\theta$. •

Theorem 7.6.4 (Completeness) *The execution of Algorithm 7.6.1 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 strongly complete and P, P_I are range-restricted, θ is a computed answer in L of $P \cup P_I \cup \{\leftarrow q(x_1, \ldots, x_k)\}$ using an SLD-refutation with the left-most-atom selection function. By Lemma 7.6.3, $q(x_1, \ldots, x_k)\theta$ is added to ans_q, which implies the assertion of the theorem. •

Theorem 7.6.5 (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 7.6.1 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 $\square$-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 7.6.3, 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 7.6.3).

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 $1 \le j < i$, 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 $1 \le k \le i-1$.

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 7.6.3. By Corollary 3.5.12, $P_I \cup \{\leftarrow \beta\}$ 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 7.6.3. By Theorem 7.6.2, $P \cup P_I \cup \{\leftarrow \beta\}$ 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 7.6.3, with $\theta = \gamma_0\delta_n$. For this, note that when Procedure 7.6.3 is called for $\leftarrow \alpha$, we have that $\alpha \in input\_p$. Besides, by Theorem 7.6.2, $P \cup P_I \cup \{\leftarrow \alpha''\}$ has an SLD-refutation in L. •

7.6.4 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 7.6.1 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 7.6.1 :

Algorithm 7.6.4

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 7.6.5 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 $\{\mathtt{t} \mid (\blacksquare, \mathtt{t}) \in ans\_q$ where $\blacksquare$ is the empty modality$\}$.

Procedure 7.6.5

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 7.6.5 to process the goal L-relation K using φ.

Procedure 7.6.6

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 7.6.5 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 α'' 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 7.6.6 *The evaluation by Algorithm 7.6.4 is sound, complete, and tight (i.e., Theorems 7.6.2, 7.6.4, 7.6.5 still hold when "Algorithm 7.6.1" is replaced by "Algorithm 7.6.4").*

Sketch. Algorithm 7.6.4 simulates Algorithm 7.6.1 but the order of calls of simulations of Procedure 7.6.2 (by using Procedure 7.6.5) is different. However, this difference does not affect the adaptation of the proofs of Algorithm 7.6.1 for Algorithm 7.6.4. •

Theorem 7.6.7 *The data complexity of Algorithm 7.6.4 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 7.2.1 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. •

7.6.5 Further Optimizations

We informally mention further optimizations for Algorithm 7.6.4:

- Step 4 of Procedure 7.6.5 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 7.6.4, 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 7.6.5 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 7.6.5 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. $\triangle'$).

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

Similarly as for the l-modal-depth-restricted fixpoint semantics, we can adjust the top-down evaluation Algorithm 7.6.4 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 7.6.2 Let $l \geq 1$ be a fixed number. We define the operator $l\text{-}rSatNF_L(J)$ 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-7.6.4 be the modification of Algorithm 7.6.4 where $rSatNF_L$ and Sat_L are replaced respectively by $l\text{-}rSatNF_L$ and $l\text{-}Sat_L$ (for Procedure 7.6.5).

It is easy to see that soundness (Theorem 7.6.2), tightness (Theorem 7.6.5), and PTIME data complexity (Theorem 7.6.7) hold for the top-down evaluation Algorithm l-7.6.4 for $L \in \mathcal{UMD}$ when l is fixed. For completeness, we have the following theorem, which can be proved similarly as Theorem 7.6.4.

Theorem 7.6.8 *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-7.6.4 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-7.6.4 really approximates top-down evaluation of L-MDatalog for $L \in \mathcal{UMD}$. Furthermore, it computes approximations with PTIME data complexity.

7.7 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 7.6.1 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 7.6.3) 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 7.6.2 and 7.6.3 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.

7.7.1 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 7.7.1 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 7.7.2 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.[5] 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

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

of predicate sup^i_j 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 7.7.3 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} :

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

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

- 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:

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

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

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

7.7.2 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 7.7.1 *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. •

$L \in \{KDI4_s5, KDI45, KDI4_s, KDI4\}$:	$\triangle input\_E \to \Box_m\, input\_E$ 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
$L = KD5$:	$\triangle input\_E \to \Box\Box input\_E$ if $\lvert\triangle\rvert \geq 1$
$L \in \{KD4, KD45\}$:	$\triangle input\_E \to \Box input\_E$ if $\lvert\triangle\rvert \geq 1$
$L \in \{S4, S5\}$:	$\triangle input\_E \to \Box input\_E$ $\triangle input\_E \to input\_E$
$L = KD4_s5_s$:	$\triangle\nabla_i\, input\_E \to \Box_i\, input\_E$
$L = KD45_{(m)}$:	$\triangle\nabla_i\triangle'\, input\_E \to \triangle\Box_i\triangle'\, input\_E$ $\triangle\nabla_i\nabla'_i\, input\_E \to \triangle\Box_i\, input\_E$
$L = sCFG$:	if $\Box_i\varphi \to \Box_{j_1} \ldots \Box_{j_k}\varphi$ is an axiom of L : $\triangle\Box_{j_1} \ldots \Box_{j_k}\triangle' input\_E \to \triangle\Box_i\triangle' input\_E$ $\triangle\nabla_i\triangle' input\_E \to \triangle\Box_{j_1} \ldots \Box_{j_k}\triangle' input\_E$
$L = KD$:	$\triangle input\_E \to \Box^h input\_E$ where $h = \lvert\triangle\rvert$
$L = T$:	$\triangle input\_E \to \Box^h input\_E$ if $h \geq \lvert\triangle\rvert - 1$
$L = B$:	$\triangle input\_E \to \Box^h input\_E$ if $h \geq \lvert\triangle\rvert - 1$ $\triangle\Box\nabla input\_E \to \Box^h input\_E$ if $h = \triangle\rvert$
$L = KDB$:	$\triangle input\_E \to \Box^h input\_E$ if $h - \lvert\triangle\rvert = 2i$ for some $i \geq 0$ $\triangle\Box\nabla input\_E \to \Box^h input\_E$ where $h = \lvert\triangle\rvert$

Table 7.1: Additional rules for specifying Sat_L

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 $rSat_L$ or rNF_L then we should accept also $\triangle input\_E \to \triangle' input\_E$, where $input\_E = input\_p(t_1, \ldots, t_k)$ if $E = p(t_1, \ldots, t_k)$. In Table 7.7.2 we give additional rules for specifying Sat_L to deal with $input\_$ atoms.

As an example, for $L = T$ and $h = \lvert\triangle\rvert$, since we have the $rSat_L$ rule $\triangle\Diamond E \leftarrow \triangle E$ and we can derive $\Box^h E$ from $\triangle E$ using a sequence of $rSat_L$ rules, we should accept the rule $\triangle\Diamond input\_E \to \Box^h input\_E$, and since $\triangle\Diamond input\_E$ is an L-instance of $\triangle\nabla input\_E$, we can

accept the rule $\triangle\nabla input\_E \to \Box^h input\_E$ with $h = |\triangle|$ for specifying Sat_L. Furthermore, for $L = T$ and $h \geq |\triangle|$, since we can derive $\Box^h E$ from $\triangle E$ using a sequence of $rSat_L$ rules, we should accept also the rule $\triangle input\_E \to \Box^h input\_E$ for specifying Sat_L. Combining the two mentioned rules, we obtain the rule $\triangle input\_E \to \Box^h input\_E$ with $h \geq |\triangle| - 1$ for specifying Sat_L.

Note that for $L \in \{KDI4_s5, KDI45, KDI4_s, KDI4, KD4I_g5_a, KD5, KD4, KD45\}$, 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 $Sat_L(\{\boxdot input\_E\})$). Similar claim can be stated for $S4$ and $S5$. 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 7.7.2 *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 7.7.3 *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 7.7.4 *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 7.7.5 *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* $\mathrm{edb}(S)$ *in L and p^γ be an adorned version of $p \in \mathrm{idb}(S)$. Then every atom $\alpha \in P_L^m(I)$ of p^γ is an L-instance of some atom from $P_L^{ad}(I)$.*

Proof. It is straightforward to prove this lemma by induction on the number of steps needed to derive α for $P_L^m(I)$, 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 7.7.6 *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* $\mathrm{edb}(S)$ *in L. Suppose that $\triangle E \in P_L^{ad}(I)$, $\boxdot$ is the universal modality being a $\Box$-lifting form of $\triangle$, and $\boxdot input\_E$ is an L-instance of some atom of $Sat_L(P_L^m(I))$. Then $\triangle E$ is an L-instance of some atom of $P_L^m(I)$.*

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 7.7.3, $\triangle'(input\_A\theta)$ is an L-instance of some atom of $Sat_L(\{\triangle input\_E\})$. By Lemma 7.7.2 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 7.7.3, 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 7.7.2, $\triangle'(B_{i_1}\theta)$ is an L-instance of some atom of $Sat_L(\{\boxdot''' E_{i_1}\})$. By Lemma 7.7.4, 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 7.7.2, $\boxdot''' input\_E_{i_1}$ 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 7.7.2, 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: $\triangle\Box_i\alpha \rightarrow \triangle\Box_j\alpha$ 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 7.7.7 *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 7.7.8 *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 7.7.1, 7.7.5, and Corollary 7.7.7. •

Chapter 8

Discussion and Conclusion

8.1 Related Works on Modal Logic Programming

In [8], Balbiani et al. gave a declarative semantics and an SLD-resolution for a class of logic programs in the monomodal logics KD, T and $S4$. To modal programs the authors associate a declarative semantics represented by a tree which is defined as the limit of a certain transformation on modal programs. The fixpoint represents a minimal Kripke model of the program. The work assumes that the $\Box$ operator does not occur in bodies of program clauses and goals. In the definition of the minimal Kripke model of a program [8], the technique of connecting each newly created world to an empty world at the time of its creation (or a similar one) is not used, hence although the minimal Kripke model of a program defined in [8] is minimal with respect to the restricted class of goals, in general it is not a least Kripke model of the program in the considered logic. There is a common point between [8] and our work: in both of the works, labeled modal operators are used to convert $\langle t\rangle(\varphi \wedge \psi)$ to $\langle t\rangle\varphi \wedge \langle t\rangle\psi$. Labeled modal operators in [8] come from Skolemization, and terms are used to label the $\Diamond$ operator. In our work, the labeling technique results from the technique of building model graphs, and we feel convenient to use classical atoms and atom variables to label $\Diamond_i$ operators.

According to the survey by Orgun and Ma [68], Akama [2] studied modal logic programming by a functional translation. In that work, modal programs are translated into programs of a two-sorted first-order logic by introducing world paths as extra parameters to function and predicate symbols. The translation method does not require the axiomatization of the accessibility relations associated with modal operators such as $\Box$ and $\Diamond$ in modal logic programs, rather it directly encodes the accessibility relations into the unification algorithm. Akama gave a meta-interpreter for executing translated programs. The modal logics considered in [2] are KD, T, KDB, B, $S4$, and $S5$. Although Akama showed that a modal Herbrand property holds for translated modal formulas, he did not give declarative semantics of modal logic programs.

In multimodal logic programming, Debart et al. [23] also used a functional translation technique. In that work, the authors gave an automated theorem proving method called Σ–E-resolution for multimodal logics. The logics have a finite number of modal operators $\Box_i$ and $\Diamond_i$ of any type among KD, KT, KD4, KT4, KF, and interaction axioms of the form $\Box_i\varphi \rightarrow \Box_j\varphi$. Modal formulas are first translated into formulas of order-sorted equational theories, preserving satisfiability, and then Σ–E-resolution is used to show satisfiability of the translated formulas. Like the work by Akama [2], extra arguments (world paths) are added to function and predicate symbols. Translated formulas are called *path formulas*. A special unification algorithm was developed to deal with path formulas and the properties of modal operators. When path formulas are restricted to Horn clauses, a logic programming language

called Pathlog is obtained. Debart et al. [23] showed that Σ–E-resolution with the restriction to Horn clauses is a sound and complete procedure for Pathlog.

The class of multimodal logics studied by Debart et al. in [23] is relatively smaller than the classes $BSMM$ and $sCFG$ considered in this work. Namely, apart from the axiom (F) : $\Box_i\varphi \equiv \Diamond_i\varphi$, the other modal axioms considered by Debart et al. in [23] are included for the classes $sCFG$ and $BSMM$, while the symmetry modal axioms (B) and (5) of $BSMM$ and interaction axioms other than (I) like $\Box_i\varphi \to \Box_j\Box_k\varphi$ are absent in the work by Debart et al. [23]. In our opinion, the approach by Debart et al. can be generalized to deal with the classes $sCFG$ and $BSMM$. However, it is not clear to us whether such an extension is straightforward or not: for example, are there only finitely many (maximally general) unifiers for any two "paths" in any $sCFG/BSMM$ logic?

Despite that Nonnengart [66] studied modal logic programming explicitly only for serial monomodal logics, his semi-functional translation method works also for serial multimodal logics. Recall that, in semi-functional translation, existential modal operators are translated away by functional translation, while universal modal operators are translated away by relational translation. As mentioned earlier, Nonnengart [66] uses accessibility relations for translated programs, but with optimized clauses for representing properties of the accessibility relations, and does not modify unification.

Another work explicitly devoted to multimodal logic programming is by Baldoni et al. [12]. The authors gave a framework for developing declarative and operational semantics for logic programs in multimodal logics which have axioms of the form $[t_1]\dots[t_n]\varphi \to [s_1]\dots[s_m]\varphi$, where $[t_i]$ and $[s_j]$ are universal modal operators indexed by terms t_i and s_j, respectively. To represent worlds in canonical models of programs, the authors used sequences of universal modal operators, which are similar to sequences of $\langle\top\rangle_i$ in our work. The work [12] contains several interesting examples (illustrating "epistemic reasoning, defining parametric and nested modules, describing inheritance in a hierarchy of classes and reasoning about actions"). The logics considered in [12] are called inclusion multimodal logics (also known as grammar logics). This class of logics is disjoint with the class of multimodal logics considered in this work. Namely, the former multimodal logics are not serial, while the latter ones are serial. However, the biggest difference between [12] and our work is that these two works base on different settings. Baldoni et al. [12] assume that modal logic programs and goals do not contain existential modal operators, while we do not adopt such a restriction. Our framework cannot cope with context-sensitive grammar logics, while the framework by Baldoni et al. [12] does not consider reasoning about possibility[1].

Our direct approach and the translation approaches all assume the conditions of seriality. With respect to the least model semantics, the semi-functional translation has the good property that it is straightforward to convert the least Herbrand model of a translated program to the least Kripke model of the original program. It seems hard to develop the least Kripke model semantics for modal logic programs using the functional translation approach. With respect to fixed/varying domain and rigid/flexible terms, Debart et al. [23] used Kripke semantics with fixed domain and rigid/flexible terms. Nonnengart [66] used Kripke semantics with varying domain and flexible terms. Baldoni et al. [12] used Kripke semantics with vary-

[1]Note that every positive propositional logic program without $\Diamond$ in $KD45$ has a least $KD45$-model with *two* possible worlds, and it cannot express complicated properties about possibility. Furthermore, existential modal operators cannot be totally replaced by universal modal operators using interaction axioms. For example, every positive propositional logic program without existential modal operators has a least $KDI4_s5$-model with $m+1$ possible worlds (recall that m is the number of different modal indices), and we have the same problem as stated before.

ing domain and rigid terms. See Garson's work [32] for a survey of the different systems for quantified modal logic.

In this work, we used Kripke semantics with fixed domain and rigid terms. This is the most common choice, but can we loose the restrictions of fixed-domain and rigid terms? Since we do not use equalities in MProlog programs, the restriction of rigid terms is not essential. What happens if we allow varying domains? First, we define a *varying-domain Kripke model* to be a tuple $M = \langle D, W, \tau, R_1, \ldots, R_m, \pi \rangle$, where for each $w \in W$, $D(w)$ is a set called the *domain of* w, $\langle W, \tau, R_1, \ldots, R_m \rangle$ is a Kripke frame, and for each $w \in W$, $\pi(w)$ is an interpretation of constant symbols, function symbols and predicate symbols on the domain $D(w)$. Second, a *variable assignment* V w.r.t. M is a function that maps each pair of a world w and a variable x to an element of the domain of w. The value of $t^{M,w}[V]$ for a term t at a world w of M is defined as usual. According to these definitions, terms are flexible. The satisfaction relation is then defined in the usual way, except that:

$$
\begin{array}{lll}
M, V, w \vDash p(t_1, \ldots, t_n) & \text{iff} & (t_1^{M,w}[V], \ldots, t_n^{M,w}[V]) \in \pi(w)(p); \\
M, V, w \vDash \forall x.\varphi & \text{iff} & \text{for all } V' \text{ different from } V \text{ only for pairs } (\_, x),\ M, V', w \vDash \varphi \\
M, V, w \vDash \exists x.\varphi & \text{iff} & \text{there exists } V' \text{ different from } V \text{ only for pairs } (\_, x) \\
 & & \text{such that } M, V', w \vDash \varphi
\end{array}
$$

Our thesis is that the framework can easily be adapted for varying-domain Kripke models. Informal argumentations for this are: First, we do not use the Barcan formula $\forall x.\Box_i\varphi \rightarrow \Box_i\forall x.\varphi$ and the converse Barcan formula $\Box_i\forall x.\varphi \rightarrow \forall x.\Box_i\varphi$ in any way. Second, as we consider only positive modal logic programs without equality, the method of constructing least Kripke models for positive modal logic programs still works for the case of varying-domain Kripke models. Precise analysis, however, should be done for this problem.

When dealing with modal logic programs with negation, the translation approaches give rise to the floundering problem[2] even when the input modal logic program and goal are *allowed*[3]. To see this, just consider the program clause $p \leftarrow \Diamond\neg q$. Extending our direct approach for dealing with negation is also a hard problem. However, we think that it is possible to overcome the difficulty and we will study this problem in the near future.

8.2 Relation between the Approaches

There are clear similarities between the functional translation approaches, the semi-functional translation approach, and the direct approach, as all of them use Kripke semantics and exploit paths in Kripke models. However, there are also differences in this aspect itself. First, the functional translation approach and the direct approach avoid to use the accessibility relations explicitly, while the semi-functional translation explicitly uses the accessibility relations for translating universal modal operators away. Second, in our direct approach and the semi-functional translation approach, "paths" form a skeleton tree of the built model graph, while in the functional translation approach, "paths" can be arbitrary in the considered Kripke model. Thus, unification of paths in our direct approach and the semi-functional translation approach can be done syntactically, while in the functional translation approach, it must be done semantically (using properties of the accessibility relations). Therefore, despite the similarities, in general the three mentioned approaches are different.

Another similarity between the approaches is that, except the work by Baldoni et al. [12], all the other works [8, 2, 23, 66, 55] and this one consider mainly only serial modal

[2] which occurs when a derived goal contains only non-ground negative literals

[3] in the sense that every variable of a clause occurs in a positive literal of the body of the clause

logics. (In [55], we studied also almost serial modal logics.) Seriality is assumed for the functional/semi-functional translation approaches because of the method used for translating existential modal operators. In our approach, seriality is required to guarantee the existence of least Kripke models and that $\Diamond_i$ is an instance of $\Box_i$.

The word "direct approach" has a relative meaning only, and one can ask how direct the approach is. In the direct approach ([8, 12] and this work), the intension is to *try* to handle modal operators directly. In general, when existential modal operators are allowed, labeling existential modal operators as in [8] and this work seems unavoidable. This means that the work [8] and this one also use some kind of "translation". However, the approach used in [8] and this work is much more "direct" than the translation approaches [2, 23, 66]. First, in [8] and this work, universal modal operators are not translated away. Second, in this work, if existential modal operators are absent in the considered program and goal (as assumed in [12]), then every SLD-derivation from those program and goal does not contain any (labeled) existential modal operator. Third, on the presentational aspect, paths in the sense of the direct approach give more flexibility than paths in the sense of the translation approaches. For example, an atom $\langle X\rangle\langle Y\rangle p(t)$ in our approach can be treated as $\langle X\rangle\,(\,\langle Y\rangle p(t)\,)$, while $p(0:x:y,t)$ in the translation approaches cannot be represented in a similar way. Similarly, $\Box\langle X\rangle p \wedge \Box\langle X\rangle q$ can be grouped as $\Box\langle X\rangle(p \wedge q)$, while such a task cannot be formulated for $p(\tau : x : f(x)) \wedge q(\tau : x : f(x))$. In our approach, we define a semantics for labeled modal operators, while in the translation approaches, it is rather not able to give a direct semantics for elements of paths.

The direct approach has a good property that it is somehow friendlier for users than the translation approaches in the debugging and iterative modes of programming. Let us consider, for example, translation of the goals $G_1 = \leftarrow \Box p$ and $G_2 = \leftarrow \Box\Diamond p(x)$. Using any of the mentioned translation methods, G_1 is translated to $\leftarrow p(\tau : a)$. The goal G_2 is translated to $\leftarrow p(\tau : f(x) : y, x)$ using the functional translation, and to $\leftarrow p(y, x), R(\tau : f(x), y)$ using the semi-functional translation. In our opinion, the translated goals are much less intuitive than the original ones. With a similar opinion, a reviewer of our conference paper [57] wrote "it is important not to translate away all modalities because the modalities allow us to separate object-level and epistemic-level notions nicely". Furthermore, if we want to let programmers to have some control in using properties of the base logic, then rules used in our approach (e.g. in the form $\triangle\Box_j\Diamond_k\alpha \leftarrow \triangle\Diamond_i\alpha$ or $\triangle\Box_i\alpha \leftarrow \triangle\Diamond_j\Box_k\alpha$) are more intuitive for them than rules used in the semi-functional translation approach (e.g. in the form $R_k(x, y) \leftarrow R_j(z, x), R_i(z, y)$).

In this work, we use classical atoms to label existential modal operators, while in the works [8, 23, 66], Skolem function symbols are used to handle existential modal operators. As observed by a reviewer, the use of modal operators indexed with atoms is akin to the use of Hilbert's epsilon operator instead of Skolem functions. Our technique has an advantage for modal deductive databases, as discussed in the next section. In the rest of this section, we argue that the labelling technique used in our direct approach is relatively better than the Skolemization technique used in the translation approaches for modal logic programming.

Consider the following logic program P in the modal logic $L = KD$:

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

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

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

The semi-functional translation of P results in $P'' = P' \cup \{R(x, x!y) \leftarrow\}$.[4] The direct consequence operator of P' has the least fixpoint $\{q(\varepsilon),\ r(\varepsilon),\ p(\varepsilon!a),\ p(\varepsilon!b)\}$. Thus, the bottom-up computation for P' or P'' gives two atoms $p(\varepsilon!a)$ and $p(\varepsilon!b)$ instead of one atom $\langle p\rangle p$ that is created by our bottom-up computation for P. As shown below, this problem corresponds to another problem in top-down computation.

Reconsider the program P and the goal G given in Example 5.5.1. Functional translation of G and P gives $G' = \leftarrow s(\varepsilon!z, x)$ and the following program P':

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

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

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

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

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

The problem is that tabulation does not work as it did in Example 5.5.1 for the direct approach, because $t(\varepsilon!g(a), a)$ is not the same as $t(\varepsilon!f(a), a)$. That is, using the functional translation approach, more computation is needed for this example.

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

[4] [23] uses '!' to construct *path expressions*, while [66] uses ':'.

and the following program P'':

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

8.3 Conclusions

One of our main contributions of this work is the framework for developing the least model semantics, fixpoint semantics, and SLD-resolution calculi for modal logic programs. The framework is formulated in a direct way (not using translation to classical logic) and closely to the style of classical logic programming. In comparison with the other works using the direct approach [8, 12], our framework does not assume any special restriction on occurrences of $\Box_i$ and $\Diamond_i$. It is applicable and useful for large classes of modal logics, including basic serial multimodal logics, serial context-free grammar logics, and the basic serial monomodal logics. The framework allows not only to exploit syntactic properties of the base logic, as in the case of $BSMM$, but also to use semantical properties of the base logic, e.g. as in the case of $KDI4_s5$. One of good features of our framework is L-normal form of modalities. In logics like $KDI4_s5$, $KDI45$, $KD4_s5_s$, $KD45_{(m)}$, $KD5$, $KD45$, $S5$, it is a tool allowing us to restrict lengths of modalities appearing in derivations. Such a tool was not introduced in [8, 2, 23, 66, 12].

In the literature of computer science, multimodal logics are much more studied for reasoning about knowledge than about belief (see, e.g., [25, 49]). In this work, we have concentrated on multimodal logics intended for reasoning about belief, in particular, for reasoning about multi-degree belief, for use in distributed systems of belief, and for reasoning about epistemic states of agents in multi-agent systems. The multimodal logic $KD4I_g5_a$ is useful for reasoning about belief and common belief of agents. The logics of multi-degree belief proposed by us are somehow similar to graded modal logics but different at the aspect that degrees in the former case are symbolic, while grades in the latter case are numeric.[5] We think that our schemata for semantics of MProlog in the considered multimodal logics of belief are practically useful. On the other hand, our schemata for semantics of $BSMM$-MProlog and $sCFG$-MProlog are interesting from the theoretical point of view. They show that declarative and procedural semantics of multimodal logic programs can be formulated in a direct way, not using translation to the classical logic. These schemata are another one of our main contributions.

We have designed and implemented MProlog to obtain high usefulness, effectiveness, and flexibility. For usefulness: codes, libraries, and most features of Prolog can be used in MProlog programs; for effectiveness: classical fragments are interpreted by Prolog itself, and a number of options can be used for MProlog to restrict the search space; for flexibility: there are three kinds of predicates (classical, modal, *dum*) and we can use and mix different calculi in an MProlog program. The MProlog system has been designed so that users can implement and add SLD-resolution calculi to the system in a modular way. We have implemented SLD-resolution calculi for a number of useful modal logics [58], including the considered multimodal

[5] Grades are used to indicate the number of worlds accessible from the current world.

logics of belief. Some of the implemented SLD-resolution calculi, e.g. the ones for KD, $KD45$, $S5$, $KDI4_s5$, $KD4_s5_s$, $KD45_{(m)}$, are quite efficient.[6]

MProlog has a different theoretical foundation than Molog. In MProlog, a labeling technique is used for existential modal operators instead of Skolemization. We also provide and use new technicalities like normal forms of modalities or pre-orders between modal operators. MProlog also eliminates drawbacks of Molog (e.g., MProlog gives computed answers).

We have given several examples demonstrating the usefulness of MProlog. The n wise men puzzle shows that *programming* is needed for a certain kind of problems, and in order to obtain elegance, programming for the n wise men puzzle should be done in modal logic.

We have also 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 like the top-down evaluation algorithm and the magic-set transformation (which can be combined with the seminaive evaluation to give a more refined bottom-up evaluation method). Our top-down evaluation algorithm for L-MDatalog queries is "tight" w.r.t. the underlying SLD-resolution calculus of L-MProlog. 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. The data complexity of L-MDatalog for $L \in \mathcal{BMD}$ is complete in PTIME and for $L \in \mathcal{UMD}$ is PSPACE-hard.[7] For $L \in \mathcal{UMD}$, we have proposed an approximation method to evaluate L-MDatalog queries in polynomial time, which works with the mentioned evaluation algorithms.

Our formulations and methods for MDatalog are highly modular w.r.t. the base modal logic L and the underlying SLD-resolution calculus for L-MProlog. This work and our previous work [54] 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 for 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 [23, 66] used in modal logic programming are not suitable for modal deductive databases, because they introduce Skolem function symbols and can make clauses not allowed.

In summary, we have developed a counterpart of the theory of classical logic programming for positive modal logic programs and modal deductive databases. Our theory uses the direct approach, is applicable for large classes of modal logics, and does not assume any special restriction on the form of logic programs, goals, and queries.

As future works, we will study SLDNF-resolution for modal logic programs with negation and try to extend MProlog with time, actions, events to deal with multi-agent systems. In our opinion, extending MProlog with concurrent dynamic logic is an interesting problem. Some temporal operators can be defined by modal operators of actions.

[6] For the mentioned logics, the *rSat* operator is either deterministic (for KD, $KD45$, and $KD4_s5_s$) or nondeterministic but with a low branching factor (2 for $KD45_{(m)}$, 3 for $S5$, and m for $KDI4_s5$).

[7] Recall that $\mathcal{BMD}$ stands for "bounded modal depth" and $\mathcal{UMD}$ stands for "unbounded modal depth".

Bibliography

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

[2] S. Akama. A meta-logical foundation of modal logic programming. 1-20-1, Higashi-Yurigaoka, Asao-ku, Kawasaki-shi, 215, Japan, December 1989.

[3] 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.

[4] C. Anger, K. Konczak, T. Linke, and T. Schaub. A glimpse of answer set programming. *Künstliche Intelligenz*, 19(1):12–18, 2005.

[5] K.R. Apt. Logic programming. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*. Elsevier, 1990.

[6] K.R. Apt and M.H. van Emden. Contributions to the theory of logic programming. *Journal of the ACM*, 29(3):841–862, 1982.

[7] G. Attardi and M. Simi. Proofs in context. In J. Doyle, E. Sandewall, and P. Torasso, editors, *KR'94: Principles of Knowledge Representation and Reasoning*, pages 16–26, San Francisco, 1994. Morgan Kaufmann.

[8] Ph. Balbiani, L. Fariñas del Cerro, and A. Herzig. Declarative semantics for modal logic programs. In *Proceedings of the 1988 International Conference on Fifth Generation Computer Systems*, pages 507–514. ICOT, 1988.

[9] Ph. Balbiani, A. Herzig, and M. Lima-Marques. TIM: The Toulouse inference machine for non-classical logic programming. In *PDK'91: International Workshop on Processing Declarative Knowledge*, pages 365–382. Springer-Verlag, 1991.

[10] Ph. Balbiani, A. Herzig, and M. Lima-Marques. Implementing Prolog extensions: A parallel inference machine. In *Proceedings of the 1992 International Conference on Fifth Generation Computer Systems*, pages 833–842. ICOT, 1992.

[11] M. Baldoni. Normal multimodal logics with interaction axioms. In D. Basin, M. D'Agostino, D.M. Gabbay, and L. Viganò, editors, *Labelled Deduction*, pages 33–57. Kluwer Academic Publishers, 2000.

[12] M. Baldoni, L. Giordano, and A. Martelli. A framework for a modal logic programming. In *Joint International Conference and Symposium on Logic Programming*, pages 52–66. MIT Press, 1996.

[13] P. Baumgartner and U. Furbach. Calculi for disjunctive logic programming. In J. Maluszynski, editor, *Proc. of ILPS 1997*, pages 229–243. MIT Press, 1997.

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

[15] A. Brogi, E. Lamma, and P. Mello. Inheritance and hypothetical reasoning in logic programming. In *Proceedings of ECAI'90*, pages 105–110, Stockholm, 1990.

[16] 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.

[17] 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.

[18] C.C. Chen and I.P. Lin. The computational complexity of the satisfiability of modal Horn clauses for modal propositional logics. *Theoretical Computer Science*, 129:95–121, 1994.

[19] J. Chomicki. Temporal query languages: A survey. In D.M. Gabbay and H.J. Ohlbach, editors, *Temporal Logic: ICTL'94*, volume 827, pages 506–534. Springer-Verlag, 1994.

[20] A. Cimatti and L. Serafini. Multi-agent reasoning with belief contexts: The approach and a case study. In M. Wooldridge and N.R. Jennings, editors, *Proceedings of ECAI-94, LNCS 890*, pages 71–85. Springer, 1995.

[21] K.L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, 1978.

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

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

[24] J.J. Elgot-Drapkin. Step-logic and the three-wise-men problem. In *AAAI*, pages 412–417, 1991.

[25] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.

[26] L. Fariñas del Cerro. MOLOG: A system that extends PROLOG with modal logic. *New Generation Computing*, 4:35–50, 1986.

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

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

[29] M. Fisher and R. Owens. An introduction to executable modal and temporal logics. In M. Fisher and R. Owens, editors, *Executable Modal and Temporal Logics, IJCAI'93 workshop*, pages 1–20. Springer, 1995.

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

[31] 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.

[32] 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.

[33] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings of the Fifth International Conference on Logic Programming*, pages 1070–1080. MIT Press, 1988.

[34] R. Goré and L.A. Nguyen. Clausal tableau systems for modal logics. Forthcoming.

[35] 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.

[36] 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.

[37] 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.

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

[39] K. Konolige. Belief and incompleteness. Technical Report 319, SRI Inter., 1984.

[40] R.A. Kowalski. Predicate logic as a programming language. *Information Processing*, 74:569–574, 1974.

[41] K. Kunen. Signed data dependencies in logic programs. *Journal of Logic Programming*, 7(3):231–245, 1989.

[42] A. Leitsch. *The Resolution Calculus*. Springer, 1997.

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

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

[45] J. Lobo, J. Minker, and A. Rajasekar. *Foundations of Disjunctive Logic Programming*. MIT Press, 1992.

[46] J. Lobo, A. Rajasekar, and J. Minker. Semantics of Horn and disjunctive logic programs. *Theoretical Computer Science*, 86(1):93–106, 1991.

[47] D. Loveland. Near-Horn Prolog. In J.-L. Lassez, editor, *Proc. of the 4th Int. Conf. on Logic Programming*, pages 456–469. MIT Press, 1987.

[48] J. McCarthy. First order theories of individual concepts and propositions. *Machine Intelligence*, 9:120–147, 1979.

[49] J.-J.Ch. Meyer and W. van der Hoek. *Epistemic Logic for Computer Science and Artificial Intelligence*. Cambridge University Press, 1995.

[50] J. Minker. Logic and databases: A 20-year retrospective. In D. Pedreschi and C. Zaniolo, editors, *Proceedings of LID'96, LNCS 1154*, pages 3–57. Springer, 1996.

[51] J. Minker and D. Seipel. Disjunctive logic programming: A survey and assessment. In A.C. Kakas and F. Sadri, editors, *Computational Logic: Logic Programming and Beyond*, volume 2407 of *LNCS*, pages 472–511. Springer, 2002.

[52] G.E. Mints. Gentzen-type systems and resolution rules. In P. Martin-Löf and G. Mints, editors, *Proceedings of COLOG-88, LNCS 417*, pages 198–231. Springer, 1988.

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

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

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

[56] L.A. Nguyen. MProlog: An extension of Prolog for modal logic programming. In B. Demoen and V. Lifschitz, editors, *Proceedings of ICLP 2004, LNCS 3132*, pages 469–470. Springer, 2004.

[57] 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.

[58] L.A. Nguyen. Source files, calculi, and examples of MProlog. Available on Internet at `http://www.mimuw.edu.pl/~nguyen/mprolog`, 2004.

[59] 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.

[60] L.A. Nguyen. An SLD-resolution calculus for basic serial multimodal logics. In D.V. Hung and M. Wirsing, editors, *Proceedings of ICTAC 2005, LNCS 3722*, pages 151–165. Springer, 2005.

[61] L.A. Nguyen. A bottom-up method for the deterministic Horn fragment of the description logic ALC. In M. Fisher et al., editor, *Proceedings of JELIA 2006, LNAI 4160*, pages 346–358. Springer-Verlag, 2006.

[62] 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.

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

[64] L.A. Nguyen. Negative ordered hyper-resolution as a proof procedure for disjunctive logic programming. *Fundamenta Informaticae*, 70(4):351–366, 2006.

[65] 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`.

[66] A. Nonnengart. How to use modalities and sorts in Prolog. In C. MacNish, D. Pearce, and L.M. Pereira, editors, *Proceedings of JELIA'94, LNCS 838*, pages 365–378. Springer, 1994.

[67] H.J. Ohlbach. A resolution calculus for modal logics. In *Proceedings of CADE-88, LNCS 310*, pages 500–516. Springer, 1988.

[68] M.A. Orgun and W. Ma. An overview of temporal and modal logic programming. In *D.M. Gabbay and H.J. Ohlbach, editors, Proc. First Int. Conf. on Temporal Logic - LNAI 827*, pages 445–479. Springer-Verlag, 1994.

[69] Y.-D. Shen, L.-Y. Yuan, J.-H. You, and N.-F. Zhou. Linear tabulated resolution based on Prolog control strategy. *TPLP*, 1(1):71–103, 2001.

[70] H. Tamaki and T. Sato. OLD resolution with tabulation. In E.Y. Shapiro, editor, *Proceedings of ICLP'1986, LNCS 225*, pages 84–98. Springer, 1986.

[71] M.H. van Emden and R.A. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM*, 23(4):733–742, 1976.

[72] A. Van Gelder, K.A. Ross, and J.S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.

[73] N.-F. Zhou and T. Sato. Efficient fixpoint computation in linear tabling. In *Proceedings of PPDP'2003*, pages 275–283. ACM, 2003.

Appendix A

More about Modules of MProlog

This appendix is a supplement to Section 6.2 on the implementation of MProlog.

A.1 Data Structures Used for MProlog

There are four main data entities used for MProlog: calculi, predicates, rules, and modal clauses. Here are predicates for manipulating data of these entities on the logical level:

For calculi:

- For setting data:
  - *dec_calculus(Calculus)* declares a new calculus and set default options for it. The parameter is the name of the declared calculus.
- For getting data:
  - *get_list_of_calculi(ListOfCalculi)* returns the list of (names of) calculi.

For predicates: (In the following, the parameter *Kind* is either *modal* or *dum*, the parameter *Predicate* stands for an element of the form Name/Arity, and the parameter *ListOfPredicates* stands for a list of elements of the form Name/Arity.)

- For setting data:
  - *dec_pred(Kind, Predicate)* declares a modal/*dum* predicate.
  - *dec_preds(Kind, ListOfPredicates)* declares a list of modal/*dum* predicates.
- For getting data:
  - *get_list_of_preds(Kind, ListOfPredicates)* returns the list of predicates of the given kind.
  - *is_pred_of_kind(Kind, Predicate)* checks if the given predicate is of the given kind.

For rules: (In the following, the parameter *Calculus* is a name and the parameter *Category* is either *pre_rSat*, *rSat*, *post_rSat*, or *rNF*.)

- For setting data:
  - *gen_rule_name(Name)* generates a new rule name.

  - *dec_mrule(Name, Rule)* declares a rule with the given name and the given content.
  - *assertz_mrule(Calculus, Category, RuleName)* adds the rule identified by *RuleName* to the list of rules of *Calculus* belonging to *Category*.
  - *set_list_of_mrules(Calculus, Category, ListOfRuleNames)* declares *ListOfRuleNames* as the list of (names of) rules of *Calculus* belonging to *Category*.

- For getting data:
  - *get_mrule(Name, Rule)* gets the content of the rule identified by *Name*.
  - *get_mrule(Name, AtomIn, PreCondition, AtomOut, PostComputation)* gets details (the head *AtomIn*, the *PreCondition*, the *atom out*, and the *PostComputation*) of the rule identified by *Name*.
  - *get_list_of_mrules(Calculus, Category, ListOfRuleNames)* gets the list of (names of) rules of *Calculus* belonging to *Category*.

For modal clauses: (In the following, the parameter *Calculus* is a name and the parameter *Predicate* stands for an element of the form Name/Arity.)

- For setting data:
  - *gen_clause_name(Name)* generates a new clause name.
  - *break_clause(Calculus, Clause, Context, Head, Body)* breaks *Clause* into *Context*, *Head*, and *Body*. The input parameter *Calculus* is used to extract the context from the clause when the body is empty.
  - *dec_mclause(Name, Context, Head, Body)* declares a clause identified by *Name* and composed by *Context*, *Head*, and *Body*.
  - *asserta_mclause(Calculus, Predicate, ClauseName)* adds the clause identified by *ClauseName* to the beginning of the list of clauses defining *Predicate* in *Calculus*.
  - *assertz_mclause(Calculus, Predicate, ClauseName)* adds the clause identified by *ClauseName* to the end of the list of clauses defining *Predicate* in *Calculus*.
  - *set_list_of_mclauses(Calculus, Predicate, ListOfClauseNames)* declares *ListOfClauseNames* as the list of (names of) clauses defining *Predicate* in *Calculus*.

- For getting data:
  - *get_mclause(Name, Context, Head, Body)* gets the context, the head, and the body of the modal clause identified by *Name*.
  - *get_mclause(Name, Context, Head, Body, BodyLength)* gets the context, the head, the body, and the body length of the modal clause identified by *Name*. The body length of a clause is the number of formulas separated by commas in the body of the clause.
  - *compose_clause(Context, Head, Body, Clause)* composes *Context*, *Head*, and *Body* into *Clause*.
  - *get_list_of_mclauses(Calculus, Predicate, ListOfClauseNames)* gets the list of clauses defining *Predicate* in *Calculus*.

On the physical level, data for calculi, predicates, rules, and modal clauses are represented by the following dynamic predicates:

- *list_of_calculi(ListOfCalculi)* keeps the list of (names of) calculi.
- *list_of_preds(Kind, ListOfPredicates)* keeps the list of all predicates (of the form Name/Arity) for each of the kinds *modal* and *dum*.
- *mrule(Name, Rule, AtomIn, PreCondition, AtomOut, PostComputation)* keeps the content and details of the rule identified by *Name*. Apart from the content (*Rule*), we keep also its computed parts to increase efficiency.
- *list_of_mrules(Calculus, Category, ListOfRuleNames)* keeps the list of (names of) rules of *Calculus* belonging to *Category*. The parameter *Category* is either *pre_rSat*, *rSat*, *post_rSat*, or *rNF*.
- *mclause(Name, Context, Head, Body, BodyLength)* keeps details of the modal clause identified by *Name*. The parameter *BodyLength* can be computed from *Body*, but we keep it for efficiency.
- *list_of_mclauses(Calculus, Predicate, ListOfClauseNames)* keeps the list of (names of) clauses defining *Predicate* in *Calculus*.

A.2 More about the Parser

A directive from a parsed file can have one of the following type:

- *pre_rSat*, *rSat*, *post_rSat*, *rNF* (for opening a section of rules),
- *calculus* (for opening a modal fragment or a classical fragment),
- *end* (for ending a section of rule or a modal fragment),
- *dum_pred* (for declaring *dum* predicates),
- *include* (for including files),
- or something else.

The names *pre_rSat*, *rSat*, *post_rSat*, *rNF*, *calculus*, and *dum_pred* are declared as unary operators with a priority weaker than ‘,’. This means that, e.g., the directive

:- pre_rSat cKDI4s5, cKD4s5s.

is treated as *‘:-’(pre_rSat((cKDI4s5, cKD4s5s)))*.

The implemented predicate for reading an element from a parsed file is

get_elem(Stream, Element, Type)

where *Element* is the content of the read element and *Type* is its type, which is one of the above given directive types, or *rule* (a rule not in a section of rules), or *clause_or_rule* (for the three other cases of clauses and rules), or *end_of_file*.

For each element read from a parsed file, depending on its type and the context, appropriate actions will be executed. We describe here such actions for each of the possible types:

- *end_of_file* : end the parsing process.
- *end* : do nothing.
- *pre_rSat*, *rSat*, *post_rSat*, *rNF* : remember the calculi of the current section of rules and open a loop to read rules until getting *end_of_file*, *end*, or a directive.

- *calculus* : remember the calculi of the fragment.
- *dum_pred* : use *dec_preds*/2 to declare the given *dum* predicates.
- *include* : recursively parse the given files.
- other types of directives: write the directive to a temporary file and consult it.
- a rule not in a section of rules: use *dec_mrule*/2 to declare the rule.
- a rule in a section of rule: use *dec_mrule*/2 to declare the rule, then call *assertz_mrule*/3 for each one of the calculi of the current section of rules in order to attach the rule to the calculi.
- a modal clause: use *break_clause*/5 to parse the clause and use *dec_mclause*/4 to declare it, then call *assertz_mclause*/3 for each one of the calculi of the current modal fragment in order to add the clause to the lists of clauses of the calculi.
- a classical clause: use *assertz*/1 to assert the clause.

A.3 Other Predicates

The interface module contains definitions of the main predicates listed in Section 6.1.3 and some friendly predicates described below:

- *mlisting(CalculusOrPredicate)* : If the argument is a modal/*dum* predicate then *mlisting* behaves similarly as *listing*/1, i.e. it displays all modal clauses defining the given predicate in every involved calculus. If the argument is a name of a calculus then *mlisting* displays all information of the calculus, i.e. options, rules, and modal clauses of the calculus.
- *get_number_of_mclauses(Calculus, Predicate, Number)* returns the number of modal clauses defining *Predicate* in *Calculus*.
- *get_modal_clause(Calculus, Predicate, Number, Clause)* returns the modal clause of the given number defining *Predicate* in *Calculus*.

The compiler contains *mcompile*/1 and three predicates with friendly names:

- *compile_loaded_calculi*/0 : This predicate is defined as *mcompile(calculi)*. It can be used in the following way: after consulting the file *mprolog.pl* to load the MProlog system, load calculi that we want to compile by using *consult_calculi*/1, then call *compile_loaded_calculi*.
- *compile_built_in_calculi*/0 : After consulting the file *mprolog.pl* to load the MProlog system, we can call this predicate to compile all the built-in calculi of the system.
- *compile_loaded_program*/0 : This predicate is defined as *mcompile(program)*. It is used in the following way: after consulting the file *mprolog.pl* to load the MProlog system, load calculi that are used for the program we want to compile by using *consult_calculi*/1, then consult the program by *mconsult*/1 and call *compile_loaded_program*.

Appendix B

Revisions of this Manuscript

Release 1.0 2006-11-02

Release 1.1 2007-03-26

1. Added: Section 3.5.3 – Strong Completeness (of SLD-resolution)
2. Added: Chapter 5 – Optimizations
3. Revised: Chapter 7 – MDatalog and Modal Deductive Databases
4. Reorganized: Appendix B of Release 1.0, which contained the proofs of correctness of the schemata for semantics of L-MProlog with $L \in \{KDI4_s,\ KDI4,\ KDI45,\ KD45_{(m)},\ KD4I_g5_a\}$, has been divided into parts and each of the parts is attached to the section directly involved with the considered logic.

Release 2.0 2008-04-21

1. Added: Section 5.4 – Restrictions, Iterative Deepening Search, and Tabulation
2. Updated: Chapter 6 – Design and Implementation of the MProlog System

Release 2.1 2009-03-06

1. Corrected: Lifting Lemma and some definitions
2. Added: Weak Lifting Lemma
3. Modified: Section 5.4 – Restrictions and Iterative Deepening Search
4. Added: Section 5.5 – Tabulation
5. Revised and extended: Section 8.2 – Relation between the Approaches

Release 2.2 2009-05-03

1. Corrected: Algorithms 7.6.1 and 7.6.4
2. Revised: Section 7.6 – Top-Down Evaluation of MDatalog