On Modal Deductive Databases*

Linh Anh Nguyen

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

Abstract. We present a query language called MDatalog, which is an extension of Datalog for multimodal deductive databases. We define modal relational algebras and give the seminaive evaluation algorithm and the magic-set transformation for MDatalog queries. Results of this paper are proved for the multimodal logics of belief $KDI4_s5$, $KDI45$, $KD4_s5_s$, $KD45_{(m)}$, which are extensions of the monomodal logic $KD45$. We show that MDatalog has PTIME data complexity in these logics.

1 Introduction

Deductive databases are very useful for practical applications. In deductive databases, intentional relations are defined using extentional relations and logical rules, and users can thus create sophisticated relations from basic ones. The field of deductive databases is mature and there are well-developed techniques for computing queries in such databases (see, e.g., [1]). Deductive databases continuously receive attention from researchers; see, e.g., recent works [2, 4].

Modal and temporal logics are used to reason about knowledge, belief, actions, changes, etc. It is desirable to study modal and temporal 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 field of temporal deductive databases has received a lot of attentions from researchers (see, e.g., the survey [3]). On the other hand, the term "modal deductive databases" is hard to find in the literature of computer science.

In [8], we proposed a modal query language MDatalog, which extends Datalog with modal operators. The computational method proposed in that work is based on building a least L-model for a modal deductive database, where L is the base modal logic. The technique used in [8] has the good property that it also works for the logics $KD4$ and $S4$ but has a disadvantage that it does not fully address advanced techniques of Datalog like the relational algebra or the magic-set transformation. In [9, 10], we developed a modal logic programming language called MProlog and gave fixpoint semantics and SLD-resolution calculi for MProlog in basic serial monomodal logics and useful multimodal logics of belief. (An implementation of MProlog was reported in [11].) We used a special

* In J. Eder, H.-M. Haav, A. Kalja, and J. Penjam, editors, Proceedings of ADBIS 2005, LNCS 3631, pages 43–57. Springer, 2005.

structure called a *model generator* to represent a Kripke model. A model generator is a set of ground modal atoms, which may contain labelled existential modal operators. The direct consequence operator of the fixpoint semantics is a function that maps a model generator to another one. With that feature, we are able to group atoms in a model generator by predicate symbols and this is a key to develop modal relational algebras, which is done in this work.

In this work, we extend the query language MDatalog for multimodal deductive databases. Basing on the existing techniques of Datalog, we define modal relational algebras and give the seminaive evaluation algorithm and the magic-set transformation for MDatalog queries. The language MDatalog is a sublanguage of MProlog and our computational methods for MDatalog are based on the fixpoint semantics of MProlog programs. Results of this paper are proved for the multimodal logics $KDI4_s5$, $KDI45$, $KD4_s5_s$, $KD45_{(m)}$, which are multimodal extensions of the monomodal logic $KD45$. The logics $KDI4_s5$ and $KDI45$ are intended for reasoning about multi-degree belief, while $KD4_s5_s$ can be used for distributed systems of belief, and $KD45_{(m)}$ can be used for reasoning about epistemic states of agents. We show that MDatalog has PTIME data complexity in these logics.

The rest of this work is organized as follows. In Section 2, we give basic definitions for multimodal logics and define the multimodal logic programming language MProlog. In Section 3, we provide fixpoint semantics for MProlog programs in the multimodal logics $KDI4_s5$, $KDI45$, $KD4_s5_s$, $KD45_{(m)}$. In Section 4, we define the MDatalog language and give definitions for multimodal deductive databases. We also show that MDatalog has PTIME data complexity in $KDI4_s5$, $KDI45$, $KD4_s5_s$, $KD45_{(m)}$. In Section 5, we define the L-SPCU algebra, which is an extension of the relational algebra SPCU for a multimodal logic L, and show that nonrecursive MDatalog queries in L can be simulated by L-SPCU queries. In Section 6, we present the seminaive evaluation algorithm and the magic-set transformation for MDatalog queries. Finally, Section 7 contains concluding remarks.

2 Preliminaries

2.1 Syntax and Semantics of Quantified Multimodal Logics

A language for quantified multimodal logics is an extension of a language of classical first-order logic with modal operators $\Box_i$ and $\Diamond_i$, for $1 \leq i \leq m$ (where m is fixed). The modal operators $\Box_i$ and $\Diamond_i$ can take various meanings. For example, $\Box_i$ can stand for "the agent i believes" and $\Diamond_i$ for "it is considered possible by agent i". The operators $\Box_i$ are called universal modal operators, while $\Diamond_i$ are called existential modal operators.

Terms and formulas are defined in the usual way, with the addition that if φ is a formula then $\Box_i\varphi$ and $\Diamond_i\varphi$ are also formulas. The *modal depth* of a formula is the maximal nesting depth of modalities in the formula. The *Herbrand universe* $\mathcal{U}$ and the *Herbrand base* $\mathcal{B}$ (for a fixed language) are defined as usual.

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

A *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. For an n-ary function symbol f, $\pi(f)$ is a function from D^n to D. For an n-ary predicate symbol p and a world $w \in W$, $\pi(w)(p)$ is an n-ary relation on D.

A *variable assignment* V w.r.t. a Kripke model M is a function that maps each variable to an element of the domain of M. The value of $t^M[V]$ for a term t is defined as usual.

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

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

and as usual for other cases (treating $\Diamond_i \varphi$ as $\neg \Box_i \neg \varphi$, and $\exists x\, \varphi$ as $\neg \forall x\, \neg \varphi$). 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 $M \vDash \alpha$ for every $\alpha \in \Gamma$.

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 a quantified multimodal logic which has a standard Hilbert-style axiomatisation denoted by $K_{(m)}$. Other *normal (multi)modal logics* are obtained by adding certain axioms to $K_{(m)}$.

For a normal modal logic L whose class of admissible interpretations can be characterized by classical first-order formulas using the accessibility relations, we call such formulas *L-frame restrictions*, and call frames with such properties *L-frames*. We call a model M with an L-frame an *L-model*. We say that φ is *L-satisfiable* if there exists an L-model of φ, i.e. an L-model satisfying φ. A formula φ is said to be *L-valid* and called an *L-tautology* if φ is true in every L-model. For a set Γ of formulas, we write $\Gamma \vDash_L \varphi$ and call φ a *logical consequence* of Γ in L if φ is true in every L-model of Γ.

2.2 Multimodal Logics about Belief

To reflect properties of belief, one can extend the system $K_{(m)}$ with some of the following axioms, where axiom (D) states that belief is consistent, axiom (I) states that subscripts indicate degrees of belief, axiom (4) (resp. (4_s)) states that belief satisfies (strong) positive introspection, and axiom (5) (resp. (5_s)) states that belief satisfies (strong) negative introspection.

Name	Schema	Corresponding Condition
(D)	$\Box_i\varphi \to \neg\Box_i\neg\varphi$	$\forall u\ \exists v\ R_i(u,v)$
(I)	$\Box_i\varphi \to \Box_j\varphi$ if $i > j$	$R_j \subseteq R_i$ if $i > j$
(4)	$\Box_i\varphi \to \Box_i\Box_i\varphi$	$\forall u,v,w\ (R_i(u,v) \land R_i(v,w) \to R_i(u,w))$
(4_s)	$\Box_i\varphi \to \Box_j\Box_i\varphi$	$\forall u,v,w\ (R_j(u,v) \land R_i(v,w) \to R_i(u,w))$
(5)	$\neg\Box_i\varphi \to \Box_i\neg\Box_i\varphi$	$\forall u,v,w\ (R_i(u,v) \land R_i(u,w) \to R_i(w,v))$
(5_s)	$\neg\Box_i\varphi \to \Box_j\neg\Box_i\varphi$	$\forall u,v,w\ (R_j(u,v) \land R_i(u,w) \to R_i(v,w))$

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

$$KDI4_s5 = K_{(m)} + (D) + (I) + (4_s) + (5)$$
$$KDI45 = K_{(m)} + (D) + (I) + (4) + (5)$$

Note that axiom (5_s) is derivable in $KDI4_s5$. 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".

For multi-agent systems, subscripts beside $\Box$ and $\Diamond$ stand for agents. For distributed systems of belief we can use $KD4_s5_s = K_{(m)} + (D) + (4_s) + (5_s)$. In this system, agents have full access to belief bases of each other. They are members of a united system and viewed as "friends". In another kind of multi-agent systems, agents are "opponents" and they play against each other. Each agent tries to simulate epistemic states of the others. To write a program for an agent one may need to use modal operators of other agents. One of suitable logics for this problem is $KD45_{(m)} = K_{(m)} + (D) + (4) + (5)$.

For further reading on modal logics, see, e.g., [6, 7].

2.3 Multimodal Logic Programs

We use E and F to denote classical atoms, and $\boxdot$ to denote a sequence of universal modal operators, which may be empty. By $\forall(\varphi)$ we denote the universal closure of φ. Similarly as in classical logic programming, we use the clausal form $\boxdot(\varphi \leftarrow \psi_1, \ldots, \psi_n)$ for $\forall(\boxdot(\varphi \lor \neg\psi_1 \ldots \lor \neg\psi_n))$.

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

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

When the base logic is intended for reasoning about multi-degree belief, it has little sense to write a program clause in the form $\Box_i\Box_j\varphi$. Besides, in the logics $KDI4_s5$ and $KD4_s5_s$ we have the tautology $\nabla\nabla'\varphi \equiv \nabla'\varphi$, where ∇ and ∇' are modal operators. For these reasons, we introduce a restriction for MProlog programs in these logics: For $L \in \{KDI4_s5, KDI45, KD4_s5_s\}$, an MProlog program is called an *L-MProlog program* if its program clauses have modal context with length bounded by 1.

In the logic $KD45_{(m)}$, we have the tautology $\Box_i\Box_i\varphi \equiv \Box_i\varphi$. So, we introduce a restriction for MProlog programs in $KD45_{(m)}$: 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 $\Box_i\Box_i$.

It is shown in [10] that the MProlog language (with goals) has the same expressiveness power as the general Horn fragment in normal modal logics. Moreover, the above restrictions do not reduce expressiveness of the language [10].

3 Fixpoint Semantics of MProlog Programs

In this section, we instantiate our framework given in [10] to provide fixpoint semantics for L-MProlog programs, where $L \in \{KDI4_s5,\ KDI45,\ KD4_s5_s,\ KD45_{(m)}\}$. Let L be one of these logics and P be an L-MProlog program.

When applying the "direct consequence operator" $T_{L,P}$, if we obtain an "atom" of the form $\Diamond_i E$, then to simplify the task we label the modal operator $\Diamond_i$. Labelling 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 $\langle E\rangle_i$.

Throughout this work, we will use the following notations:

- ∇ : $\Box_i$, $\Diamond_i$, or $\langle E\rangle_i$, called a *modal operator*;
- $\triangle$: a (possibly empty) sequence of modal operators, called a *modality*;
- A, B : formulas of the form E or ∇E, called *simple atoms*;
- α, β : formulas of the form $\triangle E$, called *atoms*;
- φ, ψ : *(labelled) formulas*, i.e. formulas that may contain $\langle E\rangle_i$.

A *ground modality* is a modality without variables.

Recall that a simple subscript like i beside $\Box$, $\Diamond$, or $\langle E\rangle$ indicates the *kind* (i.e. degree/agent number) of the modal operator. We use such subscripts beside ∇ for the same aim. To distinguish a number of modal operators we use superscripts of the form (i), e.g. $\Box^{(1)}$, $\Box^{(2)}$, $\nabla^{(i)}$.

Define that a modality $\nabla_{i_1}^{(1)} \ldots \nabla_{i_k}^{(k)}$ is in L-normal form if

- case $L \in \{KDI4_s5, KD4_s5_s\}$: $k \leq 1$,
- case $L = KD45_{(m)}$: $i_j \neq i_{j+1}$ for all $1 \leq j < k$,
- case $L = KDI45$: $i_1 > \ldots > i_k$.

A modality is in *L-normal labelled form* if it is in L-normal form and does not contain[1] $\Diamond_i$. An atom is in *L-normal (labelled) form* if it is of the form $\triangle E$ with $\triangle$ in L-normal (labelled) form. An atom is in *almost L-normal labelled form* if it is of the form $\triangle A$ with $\triangle$ in L-normal labelled form.

We define $\preceq_L$ to be the least reflexive and transitive binary relation between modal operators such that $\Diamond_i \preceq_L \langle E\rangle_i \preceq_L \Box_i$, and if $L \in \{KDI4_s5, KDI45\}$ and $i \leq j$ then $\Box_i \preceq_L \Box_j$ and $\Diamond_j \preceq_L \Diamond_i$. A ground modality $\triangle = \nabla^{(1)} \ldots \nabla^{(n)}$ is called an *L-instance* of a ground modality $\triangle' = \nabla^{(1')} \ldots \nabla^{(n')}$ if $\nabla^{(i)} \preceq_L \nabla^{(i')}$

[1] In [10], we exclude also $\langle\top\rangle_i$, but $\langle\top\rangle_i$ and $\top$ are not used in this work, as we will omit details of the construction of standard L-models of L-model generators.

for every $1 \leq i \leq n$. In that case we say that $\triangle'$ is *equal to* or *more general in L than* $\triangle$ (hereby we define a *pre-order between ground modalities*). If $\triangle$ is an L-instance of $\triangle'$ then we call $\triangle E$ an *L-instance* of the atom $\triangle' E$. For example, $\Box_1 \Diamond_2 E$ is a $KDI4_s5$-instance of $\Box_2 \langle F \rangle_1 E$.

A *model generator* is a set of ground atoms not containing $\Diamond_i$. An *L-normal model generator* is a model generator consisting of atoms in L-normal labelled form. An L-normal model generator I is expected to represent an L-model, which is defined in [10] and called the *standard L-model* of I. It is shown in [10] that *"the standard L-model of an L-normal model generator I is a least L-model of I"*.

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 $NF_L(T_{0L,P}(Sat_L(I)))$.

The *saturation operator* Sat_L is specified by the following rules, in which formulas in both sides are required to be in almost L-normal labelled form:

$$
\begin{array}{ll}
L = KDI4_s5: & \Box_i E \to \Box_j E \ \text{ if } i > j \\
& \Box_i E \to \Box_m \Box_i E \\
& \langle F \rangle_i E \to \Box_m \Diamond_i E \\
\\
L = KD4_s5_s: & \Box_i E \to \Box_j \Box_i E \\
& \langle F \rangle_i E \to \Box_j \Diamond_i E \\
\\
L = KD45_{(m)}: & \triangle \Box_i E \to \triangle \Box_i \Box_i E \\
& \triangle \langle F \rangle_i E \to \triangle \Box_i \Diamond_i E \\
\\
L = KDI45: & \triangle \Box_i \alpha \to \triangle \Box_j \alpha \ \text{ if } i > j \\
& \triangle \Box_i \alpha \to \triangle \Box_i \Box_j \alpha \ \text{ if } i > j \\
& \triangle \Box_i \Box_j \alpha \to \triangle \Box_j \alpha \ \text{ if } i > j \\
& \triangle \Box_i E \to \triangle \Box_i \Box_i E \\
& \triangle \nabla E \to \triangle \Box_i \Diamond_i E \ \text{ if } \Diamond_i \preceq_L \nabla \\
& \triangle \Box_i \nabla_j E \to \triangle \Diamond_j E \ \text{ if } i > j \\
& \triangle \langle F \rangle_i \nabla_j E \to \triangle \Diamond_i E \ \text{ if } i > j
\end{array}
$$

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 labelled form that are derivable from some atom in I using the rules specifying Sat_L. As an example, for $L = KDI4_s5$, we have $Sat_L(\{\Box_2 p(a)\}) = \{\Box_2 p(a), \Box_1 p(a), \Box_m \Box_2 p(a), \Box_m \Box_1 p(a)\}$.

Let $\boxdot$ be a universal modality in L-normal form, $\boxdot'$ a modal context of an L-MProlog program clause, φ and φ' be program clauses with an empty modal context. We say that $\boxdot$ is an *L-context instance* of $\boxdot'$ if $\boxdot' \psi \to \boxdot \psi$ is L-valid (for every ψ), and 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 $\varphi = \varphi' \theta$.

It is easily seen that $\boxdot$ is an L-context instance of $\boxdot'$ iff one of the following condition holds: a) $L \in \{KD45_{(m)}, KD4_s5_s\}$ and $\boxdot = \boxdot'$; b) $L = KDI4_s5$ and $\boxdot$ is an L-instance of $\boxdot'$; c) $L = KDI45$, $\boxdot' = \Box_i$, $\boxdot$ is not empty, and every modal operator $\Box_j$ of $\boxdot$ satisfies $j \leq i$.

The *operator* $T_{0L,P}$ is defined as follows: for a set I of ground atoms in almost L-normal labelled 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 ground modality in L-normal labelled 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 labelled form of $\triangle A$ belongs to $T_{0L,P}(I)$, where the *forward labelled form* of an atom α is the atom α' such that if α is of the form $\triangle'\Diamond_i E$ then $\alpha' = \triangle'\langle E\rangle_i E$, else $\alpha' = \alpha$.

For example, 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 $T_{0L,P}(I) = \{\langle q(a)\rangle_1\langle p(a)\rangle_1 p(a)\}$.

The *normalization operator* NF_L is specified by the following rules, in which formulas in both sides are required to be in almost L-normal labelled form and ∇_i is $\Box_i$ or $\langle E\rangle_i$:

$$
\begin{array}{ll}
L \in \{KDI4_s5, KD4_s5_s\}: & \nabla'_j\nabla_i E \to \nabla_i E \\
L = KD45_{(m)}: & \triangle\nabla'_i\nabla_i E \to \triangle\nabla_i E \\
L = KDI45: & \triangle\nabla'_j\nabla_i E \to \triangle\nabla_i E \text{ if } j \leq i
\end{array}
$$

Given a model generator I, $NF_L(I)$ is the set of all ground atoms in L-normal labelled form that are derivable from some atom of I using the rules specifying NF_L. For example, $NF_{KDI4s5}(\{\langle q(a)\rangle_1\langle p(a)\rangle_1 p(a)\}) = \{\langle p(a)\rangle_1 p(a)\}$.

Define $T_{L,P}(I) = NF_L(T_{0L,P}(Sat_L(I)))$. By definition, the operators Sat_L, $T_{0L,P}$, and NF_L are all increasingly monotonic and compact. Hence the operator $T_{L,P}$ is monotonic and continuous. By the Kleene theorem, it follows that $T_{L,P}$ 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 $n > 0$. Denote the least fixpoint $T_{L,P}\uparrow\omega$ by $I_{L,P}$ and the standard L-model of $I_{L,P}$ by $M_{L,P}$.

It is proved in [10] that *"$P \vDash_L I_{L,P}$ and $M_{L,P}$ is a least L-model of P"*.

Example 1. Consider the following program P in $L = KDI4_s5$:

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

We have $I_{L,P} = \{\langle s(a)\rangle_1\, s(a), \Box_1 r(a), \langle s(a)\rangle_1\, q(a), \Box_2 p(a), \Box_1 p(a)\}$.

4 MDatalog and Modal Deductive Databases

In this section, we give definitions for modal deductive databases and define a query language called MDatalog for such databases. We also show that the data complexity of MDatalog in the logics $KDI4_s5$, $KDI45$, $KD4_s5_s$, $KD45_{(m)}$ is in PTIME. Let L be one of these logics.

We first define the L-MDatalog language. An MProlog program clause without function symbols is *allowed* if every variable occurring in the head also occurs in the body. An *L-MDatalog program* is an L-MProlog program free from function symbols and containing only allowed clauses.

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 labelled 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 said to be in *L-normal form* if each of its tuples is of the form $(\triangle, \mathfrak{t})$ with $\triangle$ in L-normal labelled form.

A modal deductive database in L consists of an instance I of extentional L-relations (*edb*) and an L-MDatalog program P for defining intentional relations (*idb*).

If $(\triangle, \mathfrak{t})$ is a tuple in an L-relation of a predicate p then we also treat it as the atom $\triangle p(\mathfrak{t})$. Let $\mathcal{R}$ be a set of predicate symbols. An instance I of L-relations of $\mathcal{R}$ will be also treated as a set of atoms of predicates of $\mathcal{R}$. Conversely, a set I of ground atoms of predicates of $\mathcal{R}$ which are in almost L-normal labelled form will be also treated as an instance of L-relations of $\mathcal{R}$. If I is an instance of L-relations of $\mathcal{R}$ and p is a predicate symbol of $\mathcal{R}$, then by $I(p)$ we denote the instance of the L-relation p contained in I.

An L-MDatalog program P can be treated as the function P_L that maps an instance of *edb* L-relations to an instance of *idb* L-relations such that $P_L(I)$ is the least (w.r.t. $\subseteq$) L-model generator J such that $T_{L,P}(I \cup J) = J$. 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 $T_{L,P} \uparrow k$.

We define an *L-MDatalog query* to be a pair (P, φ), where P is an L-MDatalog program and $\varphi = \boxdot(query(x_1, \ldots, x_k) \leftarrow B_1, \ldots, B_h)$ is an L-MDatalog clause (i.e. an allowed program clause not containing function symbols) such that: *query* is a special predicate symbol not occurring in P and the body of φ, the variables $x_1, \ldots, x_k$ are different, and $k \geq 1$. An L-MDatalog query (P, φ) takes as input an instance I of *edb* L-relations and returns as output the L-relation $P'_L(I)(query)$, where $P' = P \cup \{\varphi\}$.

One can show that the (fixpoint) semantics of L-MDatalog queries is compatible with the least model semantics of L-MProlog programs [10].

Example 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 recognised 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 represent a branch of the company, also called an *agent*. Recall that in this logic each agent has a 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 rules 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 in $KD4_s5_s$:

agent 1:

$$\Box_1(\Diamond_1 likes(x, Coca) \leftarrow likes(x, Pepsi)) \tag{1}$$
$$\Box_1(\Diamond_1 likes(x, Pepsi) \leftarrow likes(x, Coca)) \tag{2}$$
$$\Box_1 likes(Tom, Coca) \leftarrow \tag{3}$$
$$\Box_1 likes(Peter, Pepsi) \leftarrow \tag{4}$$

agent 2:

$$\Box_2(likes(x, Coca) \leftarrow likes(x, Pepsi)) \tag{5}$$
$$\Box_2(likes(x, Pepsi) \leftarrow likes(x, Coca)) \tag{6}$$
$$\Box_2 likes(Tom, Pepsi) \leftarrow \tag{7}$$
$$\Box_2 likes(Peter, Coca) \leftarrow \tag{8}$$
$$\Box_2 likes(Peter, beer) \leftarrow \tag{9}$$

agent 3:

$$\Box_3(very\_much\_likes(x, y) \leftarrow likes(x, y), \Box_1 likes(x, y), \Box_2 likes(x, y)) \tag{10}$$
$$\Box_3 likes(Tom, Coca) \leftarrow \tag{11}$$
$$\Diamond_3 likes(Peter, Pepsi) \leftarrow \tag{12}$$
$$\Diamond_3 likes(Peter, beer) \leftarrow \tag{13}$$

for communicating with users:

$$very\_much\_likes(x, y) \leftarrow \Box_3 very\_much\_likes(x, y) \tag{14}$$
$$likes(x, y) \leftarrow \Diamond_3 very\_much\_likes(x, y) \tag{15}$$
$$possibly\_likes(x, y) \leftarrow \Diamond_i likes(x, y) \ \ (\text{for } i \in \{1, 2, 3\}) \tag{16}$$

Theorem 1. *For $L \in \{KDI4_s5, KDI45, KD4_s5_s, KD45_{(m)}\}$, the data complexity of L-MDatalog is in PTIME.*

Proof. Let (P_0, φ) be an L-MDatalog query, and I_0 an input to (P_0, φ). Let $P = P_0 \cup \{\varphi\}$, c be the size of P, and n the size of $P \cup I_0$. It is sufficient to show that the complexity of computing $P_L(I_0)$ is bounded by a polynomial of n.

Fix some $k \geq 1$ and let $I = T_{L,P,I_0} \uparrow k$ and $\alpha \in I$. Then 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 bounce by d.

The key of this proof is that modal depths of atoms appearing in $T_{L,P,I_0} \uparrow \omega$ are bounded by d. Also observe that for any atom β, the sets $Sat_L(\{\beta\})$ and $NF_L(\{\beta\})$ can be computed in a finitely bounded number of steps.

The number of classical atoms that may occur in (the atoms of) I is of rank $O(n^c)$. Hence the size of I is of rank $O(n^{c(d+1)})$. It follows that the size of $Sat_L(I_0 \cup I)$ and the number of steps needed for computing $Sat_L(I_0 \cup I)$ from I_0 and I are also of rank $O(n^{c(d+1)})$. The number of steps needed for computing $T_{0L,P}(Sat_L(I_0 \cup I))$ from $Sat_L(I_0 \cup I)$ is of rank $O(n^{c.c.(d+1)})$. The size of $T_{0L,P}(Sat_L(I_0 \cup I))$ can be estimated in a similar way as the size of I and is of rank $O(n^{c(d+2)})$. The number of steps need for computing $T_{L,P,I_0} \uparrow (k+1)$ from $T_{0L,P}(Sat_L(I_0 \cup I))$ is of the same rank as the size of $T_{0L,P}(Sat_L(I_0 \cup I))$. Therefore the number of steps needed to compute $T_{L,P,I_0} \uparrow (k+1)$ from $T_{L,P,I_0} \uparrow k$ is bounded by a polynomial of n. The size of $T_{L,P,I_0} \uparrow \omega$ can be estimated in the same way as the size of I and is of rank $O(n^{c(d+1)})$. Hence the number of steps needed to compute $T_{L,P,I_0}\uparrow\omega$ is bounded by a polynomial of n.

5 Modal Relational Algebras

Let L be one of the multimodal logics $KDI4_s5$, $KDI45$, $KD4_s5_s$, $KD45_{(m)}$. In this section, we first define a modal relational algebra in L, called the L-SPCU algebra. These algebras extend the classical SPCU algebra (see, e.g., [1]) with some operators involving with modalities. We then compare L-SPCU algebra queries with *nonrecursive* L-MDatalog programs (defined later).

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 $\geq max\{j, k\}$.

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, $\pi_{j_1,\ldots,j_n}(I) = \{(\triangle, \langle c_1, \ldots, c_n\rangle) \mid (\triangle, \mathfrak{t}) \in I$ for some $\mathfrak{t}$ with $\mathfrak{t}(j_i) = c_i$ for $1 \leq i \leq n\}$.

Cross-product This operator, denoted by $\times$, takes as input a pair of L-relations in L-normal form having arbitrary arities k and h and returns an L-relation with arity $k + h$. In particular, if $arity(I) = k$ and $arity(J) = h$, then $I \times J = \{(\triangle, \langle \mathfrak{t}(1), \ldots, \mathfrak{t}(k), \mathfrak{s}(1), \ldots, \mathfrak{s}(h)\rangle) \mid$ 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 labelled 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 union of the input relations.

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

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

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 form and returns as output an L-relation of the same arity. In particular, $\sigma_{\boxdot}(I) = \{(\triangle, \mathfrak{t}) \mid$ there exist $(\triangle', \mathfrak{t}) \in I$ and a universal modality $\boxdot'$ being an L-context instance of $\boxdot$ such that $\triangle$ is a maximal L-instance in L-normal labelled form of $\triangle'$ and $\boxdot'\}$.

Saturation This operator, denoted by Sat_L, takes as input any L-relation I in L-normal 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.

Labelling 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 $\triangle' \Diamond_i\} \cup \{(\triangle \langle p(c_1, \ldots, c_n) \rangle_i, \langle c_1, \ldots, c_n \rangle) \mid (\triangle \Diamond_i, \langle c_1, \ldots, c_n \rangle) \in I\}$.

Normalization This operator, denoted by NF_L, takes as input any L-relation I and returns as output an L-relation in L-normal 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.

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.

Observe that if input consists of finite L-relations, then the above given operations can be effectively computed and they return a finite L-relation (for $L \in \{KDI4_s5, KDI45, KD4_s5_s, KD45_{(m)}\}$).

L-SPCU (algebra) queries are built from input L-relations and unary constant relations $I^c_L = \{(\boxdot, \langle c \rangle) \mid \boxdot$ is a universal modality in L-normal labelled form$\}$, where c is a constant symbol, using the L-SPCU algebra operators.

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.

Theorem 2. *Every L-MDatalog query (P, φ), where $L \in \{KDI4_s5,\ KDI45,\ KD4_s5_s,\ KD45_{(m)}\}$ and P is a nonrecursive L-MDatalog program, is equivalent to an L-SPCU query.*

Proof. We give only a sketch for this proof. 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 $Q = \pi_1(\sigma_{1=3}(\ \Box_j(Sat_L(\sigma_{2=b}(R))) \times \Diamond_k(Sat_L(S))\ ))$. Then *ans* is equivalent to

$$NF_L(Label_{ans}(\Diamond_i^{\leftarrow}(\sigma_{\boxdot}(\ \sigma_{1=2}(Q \times Q) \times Sat_L(T) \times I^a_L\ ))))$$

The conversion of the above theorem does not hold because the operators Sat_L, $\Box_i^{\leftarrow}$ and $\Diamond_i^{\leftarrow}$ may return relations which are not in L-normal form.

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

6 Evaluation of MDatalog

In this section, we extend evaluation techniques of Datalog (see, e.g., [1]) for MDatalog. We concentrate on bottom-up techniques, in particular, the seminaive evaluation and the magic-set transformation. Specific results of this section are formulated for the logics $KDI4_s5$, $KDI45$, $KD4_s5_s$, $KD45_{(m)}$. In this section, let L denote one of these logics.

6.1 The Seminaive Evaluation

Let P be an L-MDatalog program and I an instance of *edb* L-relations. 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 $T_{L,P,I} \uparrow k$ and the content of a relation p in $T_{L,P,I} \uparrow k$ is stored in p_k. Let J_k consist of such relations p_k. Then to compute $T_{L,P,I} \uparrow (k+1)$ consider the program P_{k+1} obtained from P by replacing every predicate p in bodies of the clauses of P by p_k. P_{k+1} is a nonrecursive MDatalog program, and hence $P_{k+1}(I \cup J_k)$ can be computed using the L-SPCU algebra operators. The results of $P_{k+1}(I \cup J_k)$ 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 we can apply the *seminaive* evaluation technique in a similar way as for Datalog programs. 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.

6.2 The Magic-Set Transformation Technique

We now consider the *magic-set* transformation technique for MDatalog queries. In logic programming, SLD-resolution is a top-down procedure for computing answers. In SLD-derivations, constant symbols may be push from goals to subgoals through unification, and in this way the search space is restricted. The magic-set technique simulates that kind of search restriction for bottom-up evaluation. It rewrites a given query to another equivalent one that is more effective when used with the seminaive evaluation.

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'.

Let (P_0, φ) be an L-MDatalog query. Let ψ be the adorned version of φ w.r.t. the adornment containing only 'f' with the modification that the head is written without adornment. Let $P = P_0 \cup \{\varphi\}$ and P^{ad} be the program consisting of all adorned versions of all clauses of P_0 plus ψ. We call P^{ad} the *adorned program* corresponding to the query (P_0, φ).

We proceed by giving a further transformation for P^{ad}. 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 indexes 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[2] $\Box_i p^\gamma(t_{i_1}, \ldots, t_{i_k})$ if $\triangle = \Diamond_i$, and *input_A* otherwise. Note that we do not write adornment for *query* but it is implicitly the one that contains only 'f'. For an adorned clause $\varphi_i = \boxdot(A \leftarrow B_1, \ldots, B_k)$ and $1 \leq j \leq k$, let Sup^i_j be the atom of predicate sup^i_j whose arguments are the variables that occur both in $input\_A, B_1, \ldots, B_{j-1}$ and $B_j, \ldots, B_k, A$.

Let (P_0, φ) 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 $\boxdot'(input\_query \leftarrow)$, where $\boxdot'$ is the modal context of φ. Then for each clause $\varphi_i = \boxdot(A \leftarrow B_1, \ldots, B_k)$ of P^{ad} with the property that *query* depends on the predicate of A :

- If no *idb* predicate occurs in $B_1, \ldots, B_k$ then add to P^m the clause
  $$\boxdot(A \leftarrow input\_A, B_1, \ldots, B_k) \qquad \text{(s } i.1)$$
- Otherwise, let $i_1, \ldots, i_h$ be all the indexes such that for each $1 \leq j \leq h$, B_{i_j} is an atom of an *idb* predicate. Then add to P^m the following clauses:
  $$\begin{array}{ll} \boxdot(Sup^i_{i_1} \leftarrow input\_A, B_1, \ldots, B_{i_1-1}) & \text{(s } i.1) \\ \boxdot(Sup^i_{i_j} \leftarrow Sup^i_{i_{j-1}}, B_{i_{j-1}}, \ldots, B_{i_j-1}) \text{ for every } 1 < j \leq h & \text{(s } i.j) \\ \boxdot(A \leftarrow Sup^i_{i_h}, B_{i_h}, \ldots, B_k) & \text{(s } i.(h+1)) \\ \boxdot(input\_blf\_B_{i_j} \leftarrow Sup^i_{i_j}) \text{ for every } 1 \leq j \leq h & \text{(i } i.j) \end{array}$$

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

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

Among the clauses of P^m there is exactly one clause defining *query*. Denote that clause by φ^m. Then (P^m, φ^m) is the L-MDatalog query obtained from (P_0, φ) by the magic-set transformation.

In order to compare (P^m, φ^m) with (P_0, φ) and obtain an equivalence we need a modification for the operator Sat_L. The problem is that if $\triangle E \to \triangle' E$ is an instance of a rule specifying Sat_L or NF_L then we should accept also $\triangle' \, input\_E \to \triangle input\_E$. We extend the primary set of rules specifying Sat_L with the following rules:

– case $L \in \{KDI4_s5, KDI45\}$: $\triangle input\_E \to \Box_m \, input\_E$ if $|\triangle| \geq 1$;
– case $L = KD4_s5_s$: $\triangle\nabla_i \, input\_E \to \Box_i \, input\_E$;
– case $L = KD45_{(m)}$: $\triangle\nabla_i\triangle' \, input\_E \to \triangle\Box_i\triangle' \, input\_E$ and $\triangle\nabla_i\nabla'_i \, input\_E \to \triangle\Box_i \, input\_E$.

We also need the modification that if α is of the form $\triangle\Diamond_i \, p^\gamma(t_1, \ldots, t_n)$ then the *forward labelled form* of α is $\triangle\langle p(t_1, \ldots, t_n)\rangle_i \, p^\gamma(t_1, \ldots, t_n)$ instead of $\triangle\langle p^\gamma(t_1, \ldots, t_n)\rangle_i \, p^\gamma(t_1, \ldots, t_n)$. The following theorem states that the magic-set transformation for L-MDatalog is correct. See [10] for its proof.

Theorem 3. *Let $L \in \{KDI4_s5,\ KDI45,\ KD4_s5_s,\ KD45_{(m)}\}$, (P_0, φ) be an L-MDatalog query, $P = P_0 \cup \{\varphi\}$, (P^m, φ^m) be the result of the magic-set transformation for (P_0, φ), and I an* edb *instance. Then every atom $\triangle query(c_1, \ldots, c_k) \in P^m_L(I)$ is an L-instance of some atom of $P_L(I)$, and every atom $\triangle query(c_1, \ldots, c_k) \in P_L(I)$ is an L-instance of some atom of $P^m_L(I)$. This means that $RE_L(P^m_L(I)(query)) = RE_L(P_L(I)(query))$.*

Given an L-MDatalog query (P_0, φ), to evaluate it we can first transform it into (P^m, φ^m) using the magic-set transformation, and then apply the seminaive evaluation for the new query.

7 Conclusions

In this work, we have presented the modal query language MDatalog and developed modal relational algebras and evaluation methods for MDatalog queries. We have applied our methods for the multimodal logics of belief $KDI4_s5$, $KDI45$, $KD4_s5_s$, $KD45_{(m)}$ and shown that MDatalog has PTIME data complexity in these logics.

Our methods are applicable for other modal logics. In particular, they can be applied for the serial modal logics *KD*, *T*, *KDB*, *B*, *KD*5, *S*5, and extended for the almost serial modal logics *KB*, *K*5, *K*45, and *KB*5, as fixpoint semantics for MProlog programs in these logics have been developed in [9].

Looking from the view of modal logic programming, the ability of adopting the fixpoint semantics of MProlog programs for computing MDatalog queries is an evidence for the usefulness of the direct approach used for modal logic programming [9, 10]. The translational approaches [5, 12] 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 the field of deductive databases, apart from bottom-up methods like the seminaive evaluation or the magic-set transformation, there are also top-down methods. It is known that, for Datalog, the magic-set transformation method is "equivalent" to the top-down QSQ method (see, e.g., [1]). For MDatalog, we did not give any top-down method for evaluation. This remains as an interesting problem for further investigation, at least from the theoretical point of view. There are also other problems deserving for investigation, e.g., behaviours of the redundant elimination operator, efficient representation of *edb* databases, or further optimisations for bottom-up evaluation methods.

This work and our previous work [8] are pioneer works on modal deductive databases. Despite that this work does not cover all problems involving with modal deductive databases, it establishes a fundamental basis for the subject.

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

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* Addison Wesley, 1995.
2. A. Behrend and R. Manthey. Update propagation in deductive databases using soft stratification. In G. Gottlob, A.A. Benczúr, and J. Demetrovics, editors, *Proceedings of ADBIS'04, LNCS 3255*, pages 22–36. Springer, 2004.
3. 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.
4. C. Cumbo, W. Faber, G. Greco, and N. Leone. Enhancing the magic-set method for disjunctive datalog programs. In B. Demoen and V. Lifschitz, editors, *Proceedings of ICLP'04, LNCS 3132*, pages 371–385. Springer, 2004.
5. F. Debart, P. Enjalbert, and M. Lescot. Multimodal logic programming using equational and order-sorted logic. *Theoretical Computer Science*, 105:141–166, 1992.
6. M. Fitting and R.L. Mendelsohn. *First-Order Modal Logic.* Springer, 1998.
7. J.Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artif. Intell.*, 54(2):319–379, 1992.
8. L.A. Nguyen. The modal query language MDatalog. *Fundamenta Informaticae*, 46(4):315–342, 2001.
9. L.A. Nguyen. A fixpoint semantics and an SLD-resolution calculus for modal logic programs. *Fundamenta Informaticae*, 55(1):63–100, 2003.
10. L.A. Nguyen. Multimodal logic programming and its applications to modal deductive databases. Manuscript (served as a technical report), available on Internet at `http://www.mimuw.edu.pl/~nguyen/papers.html`, 2003.

11. 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.
12. 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.