*Fundamenta Informaticae 46 (2001) 315–342*
*IOS Press*


# The Modal Query Language MDatalog

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

**Abstract.** We propose a modal query language called MDatalog. A rule of an MDatalog program is a universally quantified modal Horn clause. This language is interpreted in fixed-domain first-order modal logics over signatures without functions. We give algorithms to construct the least models for MDatalog programs. We show PTIME complexity of computing queries for a given MDatalog program in the logics $KD$, $T$, $KB$, $KDB$, $B$, $K5$, $KD5$, $K45$, $KD45$, $KB5$, and $S5$, provided that the quantifier depths of queries and the program are finitely bounded, and that the modal depth of the program is finitely bounded in the case when the considered logic is not an extension of $K5$. Some examples are given to illustrate application of the techniques to reason about belief and knowledge.

## 1. Introduction

Deductive databases and logic programming are fields that have gained a great deal of attention due to their usefulness and a wide range of applications. These fields are closely related to each other, they both use classical logic rules to represent mutual dependencies between relations. While logic programming is mostly used in artificial intelligence systems, deductive databases are systems in which the amount of data is much greater than the number of rules. Primarily, logic rules in such systems are restricted to the positive form, i.e. to universally quantified Horn clauses of the classical first-order logic. It is known that with such a restriction, for the case when the signature does not contain function symbols, the least Herbrand model for a program (or a database) is computable in polynomial time and its size is bounded by a polynomial. For surveys of the fields see the works [1, 2, 8].

In the classical propositional and first-order logics we are not able, or hardly able, to reason about necessity, belief, knowledge, etc. In order to deal with such modalities it is natural to use modal logics.

*Address for corresponding: Institute of Informatics, University of Warsaw, ul. Banacha 2, 02-097, Warsaw, Poland

In modal logics we have two additional unary modal connectives $\Box$ and $\Diamond$, which may have different meanings in different logics, for example, $\Box$ can stand for "necessity", and $\Diamond$ for "possibility".

When trying to extend the fields of deductive databases and logic programming to deal with modal logics, a natural attempt is to consider the modal Horn fragment. In [3] Fariñas del Cerro and Penttonen showed that the problem of checking the satisfiability of a set of modal Horn clauses in $S5$ is decidable in PTIME. In [5] Chen and Lin showed that the similar problem for a normal modal logic $L$ which is an extension of $K5$ is also decidable in PTIME. Chen and Lin also proved that satisfiability of a set of modal Horn clauses in the modal logics *K*, *KD*, *T*, *KB*, *KDB*, *B*, *K*4, *KD*4, and *S*4 is PSPACE-complete. The modal Horn fragment, in those works, was only studied in the context of top-down methods, which are suitable for logic programming but not deductive databases. With bottom-up methods, we often deal with constructing the least models for positive programs. Having the least model of a positive program, all queries to that program can be effectively answered.

In our recent work [9], we formulated so called positive propositional modal logic programs, which have a close relationship with the modal Horn fragment. For any set $X$ of Horn clauses, in any propositional normal modal logic *L*, there exists a positive program $P$ and a query $\phi$ (i.e. a positive formula) such that $X$ is *L*-satisfiable iff $\phi$ does not follow from $P$ in *L*. In the mentioned work, we gave algorithms to construct the least $L$-model for a given positive propositional modal logic program $P$, where $L$ can be one of the propositional modal logics $KD$, $T$, $KDB$, $B$, $KD4$, $S4$, $KD5$, $KD45$, and $S5$. If $L \in \{KD5, KD45, S5\}$, or $L \in \{KD, T, KDB, B\}$ and additionally the modal depth of $P$ is finitely bounded, then the least $L$-model of $P$ can be constructed in PTIME and coded in polynomial space. For the modal logics $KB$, $K5$, $K45$, and $KB5$, we showed that $P$ can be characterized (see Definition 2.13) in such logics by one or two models, which can be effectively constructed (in PTIME when $L \in \{K5, K45, KB5\}$).

This paper is a continuation of the above mentioned work. We extend positive propositional modal logic programs to the first-order version by allowing universally quantified program rules. Thus a modal logic query language called *MDatalog* is defined. This language is interpreted in fixed-domain first-order modal logics over signatures without functions. An MDatalog program can be treated as the set of all of propositional instances of its rules. This gives a straightforward method to estimate complexity of constructing the least models for MDatalog programs. However, some other strategies discussed in this work will result in more efficient algorithms.

Moreover, we show PTIME complexity of computing queries for a given MDatalog program in the logics $KD$, $T$, $KB$, $KDB$, $B$, $K5$, $KD5$, $K45$, $KD45$, $KB5$, and $S5$, provided that the quantifier depths of queries and the program are finitely bounded, and that the modal depth of the program is finitely bounded in the case when the considered logic is not an extension of $K5$.

This work is organized as follows: In Section 2 we give basic definitions for propositional modal logics, including the definition of positive propositional modal logic programs. In Section 3 we recall the results of [9] on constructing the least models for positive propositional modal logic programs. We recall in Section 4 the definition of fixed-domain first-order modal logics over signatures without functions. The modal query language MDatalog is defined in Section 5. In Section 6 we define an order between first-order Kripke models. Existence of the least models of MDatalog programs and computational complexity for answering queries are analyzed in Section 7. In Section 8 we discuss how algorithms of constructing the least models for MDatalog programs can be implemented, and propose a technique that optimizes both execution time and memory. Some examples of application are given in Section 9.

# 2. Definitions for Propositional Modal Logics

## 2.1. Syntax and Semantics

The sentences of propositional modal logics are built from primitive propositions $p_1, p_2, \ldots$, classical connectives $\wedge$, $\vee$, $\neg$, $\rightarrow$, nonclassical unary modal connectives $\Box$, $\Diamond$, and parentheses ), (.

A propositional modal formula, hereafter simply called a *formula*, is any finite sequence of these symbols obtained by applying the following rules: any proposition $p_i$ is a formula, and if $\phi$ and $\psi$ are formulae then so are $\neg\phi$, $\phi \wedge \psi$, $\phi \vee \psi$, $\phi \rightarrow \psi$, $\Box\phi$, and $\Diamond\phi$.

In this and the next section, the adjective "propositional" is sometimes omitted, as we are talking only about propositional modal logics.

The symbols $\neg$, $\wedge$, $\vee$ and $\rightarrow$, respectively, stand for logical negation, logical conjunction, logical disjunction and logical (material) implication. The symbols $\Box$ and $\Diamond$ can take various meanings but traditionally stand for "necessity" and "possibility".

We use letters $p$ and $q$ to denote primitive propositions, and Greek letters $\phi$, $\psi$, $\zeta$ to denote formulae. We denote the set of primitive propositions by $\mathcal{P}$, and the set of formulae by $\mathcal{F}$.

**Definition 2.1. (Kripke Frames)**
A *Kripke frame* is a triple $\langle W, \tau, R\rangle$, where $W$ is a nonempty set of possible worlds, $\tau \in W$ is the actual world, and $R$ is a binary relation on $W$, called the *accessibility relation*. If $R(w, u)$ holds then we say that the world $u$ *is accessible from the world* $w$, or that $u$ *is reachable from* $w$.

If $R$ is a binary relation then by $R^*$ we denote its transitive closure. A frame $\langle W, \tau, R\rangle$ is said to be *connected* if every world from $W$ is directly or indirectly reachable from $\tau$ (i.e. $\forall x \in W\ x = \tau \vee R^*(\tau, x)$).

**Definition 2.2. (Kripke Models)**
A *Kripke model* is a tuple $\langle W, \tau, R, h\rangle$, where $\langle W, \tau, R\rangle$ is a Kripke frame, $h : W \rightarrow P(\mathcal{P})$, and $h(w)$ is the set of primitive propositions which are "true" at the world $w$.

**Definition 2.3. (Model Graphs)**
A *model graph* is a tuple $\langle W, \tau, R, H\rangle$, where $\langle W, \tau, R\rangle$ is a Kripke frame, $H : W \rightarrow P(\mathcal{F})$, and $H(w)$ is the set of formulae which are expected to be "true" at the world $w$.

We sometimes treat model graphs as models with $H$ being restricted to the set of primitive propositions.

**Definition 2.4. (Satisfiability)**
Given some Kripke model $M = \langle W, \tau, R, h\rangle$, some world $w \in W$, the satisfaction relation $\models$ is defined as follows:

$$
\begin{array}{lll}
M, w \models p & \text{iff} & p \in h(w); \\
M, w \models \neg\phi & \text{iff} & M, w \not\models \phi; \\
M, w \models \phi \wedge \psi & \text{iff} & M, w \models \phi \text{ and } M, w \models \psi; \\
M, w \models \phi \vee \psi & \text{iff} & M, w \models \phi \text{ or } M, w \models \psi; \\
M, w \models \phi \rightarrow \psi & \text{iff} & M, w \not\models \phi \text{ or } M, w \models \psi; \\
M, w \models \Box\phi & \text{iff} & \text{for all } v \in W \text{ such that } R(w, v),\ M, v \models \phi; \\
M, w \models \Diamond\phi & \text{iff} & \text{there exists some } v \in W \text{ such that } R(w, v) \text{ and } M, v \models \phi.
\end{array}
$$

We say that $M$ *satisfies* $\phi$ *at* $w$ iff $M, w \vDash \phi$, and that $M$ *satisfies* $\phi$, or $\phi$ is *satisfied* in $M$, and write $M \vDash \phi$, iff $M, \tau \vDash \phi$.

**Definition 2.5. (Size of a Kripke Model)**
The *size* of a Kripke model $M = \langle W, \tau, R, h \rangle$ (resp. a model graph $M = \langle W, \tau, R, H \rangle$) is the sum of the number of its worlds, the size of its accessibility relation, and the total number of primitive propositions (resp. formulae) from its worlds, i.e. $|W| + |R| + \Sigma_{w \in W} |h(w)|$ (resp. $|W| + |R| + \Sigma_{w \in W} |H(w)|$).

**Definition 2.6. (Length and Modal Depth of a Formula)**
The *length* of a formula $\phi$ is defined to be the number of connectives and primitive propositions in $\phi$. The *modal depth* of a formula $\phi$, denoted by $mdepth(\phi)$, is the maximal nesting depth of modalities occurring in $\phi$, i.e.:

$$\begin{aligned}
&mdepth(p) = 0; \;\; mdepth(\neg\phi) = mdepth(\phi); \\
&mdepth(\phi \wedge \psi) = mdepth(\phi \vee \psi) = \\
&mdepth(\phi \to \psi) = max(mdepth(\phi), mdepth(\psi)); \\
&mdepth(\Box\phi) = mdepth(\Diamond\phi) = mdepth(\phi) + 1.
\end{aligned}$$

**Lemma 2.1.** Given a model $M$ and a formula $\phi$, the problem of checking whether $M \vDash \phi$ is solvable in polynomial time (in the size of $M$ and the length of $\phi$).

The proof of this lemma is straightforward, and it can be found in [9].

**Definition 2.7. (Depth of a World)**
For a Kripke frame $\langle W, \tau, R \rangle$, for $w \in W$, $w \neq \tau$, we define the *depth* of $w$ to be the smallest number $k$ such that $R^k(\tau, w)$. The depth of $\tau$ is assumed to be 0. The depth of a world that is not directly or indirectly reachable from $\tau$ is undefined. We denote the depth of $x$ by $depth(x)$.

**Definition 2.8. (Restricted Models)**
Let $M$ be a Kripke model. For $k \geq 0$, we define $M|_k$ to be the model obtained from $M$ by restricting it to the worlds with depth not greater then $k$.

**Lemma 2.2.** Let $M$ be a Kripke model. A formula $\phi$ with modal depth $m$ is satisfied in $M$ iff it is satisfied in $M|_m$.

This lemma can be easily proved by induction on the construction of $\phi$.

## 2.2. Modal Logic Correspondences

The simplest normal modal logic (called *K*) is axiomatized by the standard axioms for the classical propositional logic, the *modus ponens* inference rule, the *K-axiom* schema $\Box(\phi \to \psi) \to (\Box\phi \to \Box\psi)$, plus the additional *necessitation rule*

$$\frac{\vdash \phi}{\vdash \Box\phi}$$

It can be shown that a modal logic formula is provable in this axiomatization iff it is satisfied in every Kripke model (i.e. without any special $R$-properties) [7]. It is known that certain axiom schemata added to this axiomatization correspond to certain properties of the accessibility relation (see also [6, 4]).

| Axiom | Schemata | First-Order Formula |
|---|---|---|
| D | $\Box\Phi \to \Diamond\Phi$ | $\forall w\, \exists u\, R(w, u)$ |
| T | $\Box\Phi \to \Phi$ | $\forall w\, R(w, w)$ |
| B | $\Phi \to \Box\Diamond\Phi$ | $\forall w, u\, R(w, u) \to R(u, w)$ |
| 4 | $\Box\Phi \to \Box\Box\Phi$ | $\forall w, u, v\, R(w, u) \land R(u, v) \to R(w, v)$ |
| 5 | $\Diamond\Phi \to \Box\Diamond\Phi$ | $\forall w, u, v\, R(w, u) \land R(w, v) \to R(u, v)$ |

Table 1. Axioms and corresponding conditions on $R$

Many of such correspondences are definable as formulae of first-order logic where the binary predicate $R(x, y)$ represents the accessibility relation, as shown in Table 1. Different modal logics are distinguished by their respective additional axiom schemata. Some of the most popular modal logics together with their axiom schemata are listed in Table 2. We refer to the properties of the accessibility relation of a modal logic *L* as the *L-frame axioms* or *L-frame restrictions*.

**Definition 2.9. ($L$-satisfiability)**
We call a model $M$ a *L-model* if the accessibility relation of $M$ satisfies all *L*-frame restrictions. We say that $\phi$ is *L-satisfiable* if there exists an *L*-model of $\phi$, i.e. an *L*-model satisfying $\phi$. A formula $\phi$ is a *tautology* in a logic *L* if $\phi$ is satisfied in every *L*-model. We write $\phi \vDash_L \psi$ to denote that $\psi$ is satisfied in every *L*-model of $\phi$.

By *normal modal logics* we call logics that are extensions of the logic *K*. Let *L* be one of the modal logics listed in Table 2. For a binary relation $R$, we use $Ext_L(R)$ to denote the least extension of $R$ that satisfies all *L*-frame axioms, excluding the axiom D. It is clear that this operator is well defined for such logics.

## 2.3. Positive Modal Logic Programs

In this section, we give the definition of positive propositional modal logic programs. There is a close relationship between positive modal logic programs and *Horn formulae* (see Definition 2.11).

A formula is in the *negative normal form* if it does not contain the connective $\to$, and each of its negations occurs immediately before a primitive proposition. Every formula can be transformed to the equivalent negative normal form in a usual way.

**Definition 2.10. (Negative and Positive Formulae)**
A formula is called *negative* if in its negative normal form every primitive proposition is prefixed by negation. A formula is called *positive* if its negation is a negative formula.

**Definition 2.11. (Modal Horn Formulae)**
A formula $\phi$ is called a *Horn formula* iff one of the following conditions holds:

- $\phi$ is a primitive proposition;
- $\phi$ is a negative formula;

| Logic | Axioms | Frame Restriction |
|---|---|---|
| *K* | K | no restriction |
| *KD* | KD | serial |
| *T* | KT | reflexive |
| *KB* | KB | symmetric |
| *KDB* | KDB | serial and symmetric |
| *B* | KTB | reflexive and symmetric |
| *K*4 | K4 | transitive |
| *KD*4 | KD4 | serial and transitive |
| *S*4 | KT4 | reflexive and transitive |
| *K*5 | K5 | euclidean |
| *KD*5 | KD5 | serial and euclidean |
| *K*45 | K45 | transitive and euclidean |
| *KD*45 | KD45 | serial, transitive and euclidean |
| *KB*5 | KB5 | symmetric and euclidean |
| *S*5 | KT5 | reflexive and euclidean |

Table 2. Modal logics and frame restriction

- $\phi = \Box\psi$, or $\phi = \Diamond\psi$, or $\phi = \psi \wedge \zeta$, where $\psi$ and $\zeta$ are Horn formulae;
- $\phi$ is a disjunction of a negative formula and a Horn formula.

**Definition 2.12. (Positive Modal Logic Programs)**
A *positive propositional modal logic program* is a finite set of rules of the following form:

$$\Box^s(B_1 \wedge \ldots \wedge B_k \to A)$$

where $s \geq 0$, $k \geq 0$, and $A, B_1, \ldots, B_k$ are formulae of one of the forms $p$, $\Box p$, $\Diamond p$, where p is a primitive proposition. We often write program rules in the form:

$$\Box^s[A \leftarrow B_1, \ldots, B_k].$$

**Definition 2.13. (Characterizing Positive Programs by Models)**
We say that a positive program $P$ is *characterized* in a normal modal logic $L$ by a set $\mathcal{M}$ of models if $\mathcal{M}$ contains only $L$-models of $P$, and for any positive formula $\phi$, $P \models_L \phi$ iff $\phi$ is satisfied in every model of $\mathcal{M}$.

A set of formulae is sometimes also considered as the conjunction of its formulae, in particular when we are talking about length, modal depth, or satisfiability. The following proposition concerns the relationship between positive modal logic programs and Horn formulae. It is similar to one given in [9].

**Proposition 2.1.** For any set $X$ of Horn formulae there exists a positive modal logic program $P$ and a positive formula $\phi$ such that:

- $X$ is $L$-satisfiable iff $P \not\models_L \neg\phi$, where $L$ is any normal modal logic.
- The modal depth of the set $P \cup \{\phi\}$ is equal to the modal depth of $X$, and its length is of quadratic order in the length of $X$.

The translation from $X$ to $P$ and $\phi$ is solvable in polynomial time.

This proposition shows that our definition of positive modal logic programs is not as restrictive as it seems. It is also a connection to study the complexity of the satisfiability problem of sets of Horn formulae via the complexity of positive modal logic programs. As a corollary of Theorems 3.2, 3.3 and 3.4, the satisfiability problem of sets of Horn formulae in the logics *K*5, *KD*5, *K*45, *KD*45, *KB*5, and *S*5 is decidable in PTIME (see also [5, 3, 9]), the satisfiability problem of sets of Horn formulae with finitely bounded modal depth in the logics *KD*, *T*, *KB*, *KDB*, and *B* is also decidable in PTIME (see also [9]).

# 3. Results on Propositional Modal Logic Programming

In this section, we briefly recall our results of [9] on constructing the least models for positive propositional modal logic programs. We give only the algorithms and omit the proofs. The algorithms will further be extended to deal with the first-order case.

## 3.1. Ordering Kripke Models

In this subsection, we define an order between Kripke models. This order has the property that if a model $M$ is less than or equal to $N$, then for any positive formula $\phi$, if $\phi$ is satisfied in $M$ then $\phi$ is also satisfied in $N$. Using the order we then define the least $L$-model of a positive modal logic program.

**Definition 3.1. (Order between Kripke Models)**
Let $M = \langle W, \tau, R, h\rangle$ and $N = \langle W', \tau', R', h'\rangle$ be Kripke models. We say that $M$ is less than or equal to $N$ wrt. a binary relation $r \subseteq W \times W'$, and write $M \leq N$ wrt. $r$, if the following conditions hold:

1. $r(\tau, \tau')$
2. $\forall x, x' \;\; r(x, x') \to (h(x) \subseteq h'(x'))$
3. $\forall x, x', y \;\; R(x, y) \wedge r(x, x') \to \exists y' \; R'(x', y') \wedge r(y, y')$
4. $\forall x, x', y' \;\; R'(x', y') \wedge r(x, x') \to \exists y \; R(x, y) \wedge r(y, y')$.

We say that a model $M$ is less than or equal to $N$, and write $M \leq N$, if $M \leq N$ wrt. some $r$.

In the above definition, $r(x, x')$ intuitively states that the world $x$ is less than or equal to $x'$. The relation $\leq$ between Kripke models is a pre-order (see [9]).

**Lemma 3.1.** If $M \leq N$ wrt. $r$, then for every positive formula $\phi$ and for every $u$ and $u'$ such that $r(u, u')$, if $M, u \models \phi$ then $N, u' \models \phi$. In particular, if $M \leq N$ then for every positive formula $\phi$, if $M \models \phi$ then $N \models \phi$.

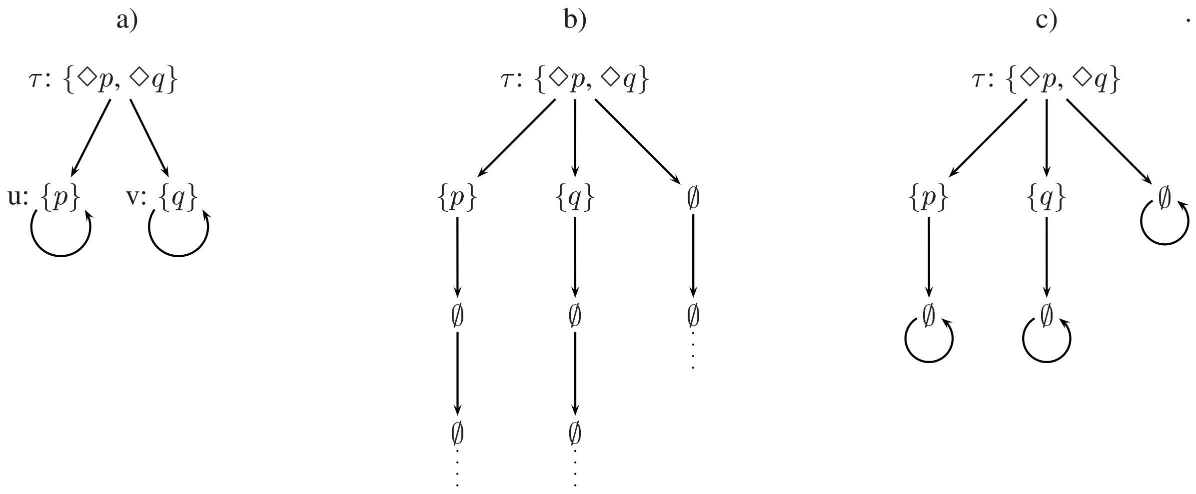


Figure 1. a) A *KD*-model of $P = \{\Diamond p, \Diamond q\}$, b) The least *KD*-model of $P$, c) A *KD*-model equivalent to the model in b).

**Definition 3.2. (The Least $L$-Model of a Positive Program)**
Let $P$ be a positive program in a normal modal logic *L*. We say that $M$ is the least *L*-model of $P$ if $M$ is an *L*-model of $P$ and $M$ is less than or equal to every *L*-model of $P$.

Observe that if $P$ is a positive program in a normal modal logic *L*, and $M$ is the least *L*-model of $P$, then for any positive formula $\phi$, $M \models \phi$ iff $P \models_L \phi$.

### 3.2. Constructing the Least Models for Positive Programs in *KD*, *T*, *KDB*, *B*, *KD*5, *KD*45, and *S*5

In this subsection we give an algorithm to construct the least $L$-model for a given positive modal logic program $P$, where $L$ can be one of the modal logics $KD$, $T$, $KDB$, $B$, $KD5$, $KD45$, and $S5$.

To illustrate the idea of the algorithm, let us consider an example. Let $P = \{\Diamond p, \Diamond q\}$ be a positive program. $M = \langle\{\tau, u, v\}, \tau, R, h\rangle$, illustrated in Figure 1a, with $R = \{(\tau, u), (\tau, v), (u, u), (v, v)\}$, $h(\tau) = \{\}$, $h(u) = \{p\}$, and $h(v) = \{q\}$, is a *KD*-model of $P$. But $M$ is not the least *KD*-model of $P$. Why not? One of the reasons is that $M \models \Box(p \lor q)$ but $\Diamond p \land \Diamond q \not\models_{KD} \Box(p \lor q)$. How can we modify $M$ to make it the least *KD*-model of $P$? First, connect $\tau$ to a new empty world, and this new world to itself (in order to satisfy the *KD*-frame restriction). $M$ then no longer satisfies $\Box(p \lor q)$. But it still satisfies, for example, $\Diamond\Box p$, which does not follow from $\Diamond p \land \Diamond q$ in *KD*. Now connect $u$ to a new empty world, and note that $M$ no longer satisfies $\Diamond\Box p$. Generalizing these observations, one can hope that connecting each newly created world to an empty world is the key to construct the least models for positive modal logic programs. Figure 1b represents the least *KD*-model of $P$.

#### 3.2.1. The Case for the Logics *KD*, *T*, *KDB*, and *B*

Let $L$ be one of the modal logics *KD*, *T*, *KDB*, *B*. The least $L$-model for $P$ will be constructed by building a $L$-model graph for $P$. Formulae from the contents of worlds of the model graph are treated as requirements to be realized. We will realize formulae of the form $(A \leftarrow B_1, \ldots, B_k)$ or $\Box\psi$ in a usual way (see Algorithm 3.1). To realize a formula $\Diamond\psi$ at a world $u$ we connect $u$ to a new world containing $\psi$. To guarantee the model graph to be the smallest, we connect each newly created world to an infinite chain of empty worlds. From time to time, we extend the accessibility relation in the way to satisfy the $L$-frame restrictions.

In the algorithm given below, as a data structure we have a model graph $M = \langle W, \tau, R, H \rangle$. We sometimes refer to $M$ as the *model* $\langle W, \tau, R, h \rangle$, with $h(x) = \{p : p$ is a primitive proposition belonging to $H(x)\}$. We write $M, u \vDash \phi$ to mean that the formula $\phi$ is satisfied at the world $u$ in the *model* $M$.

We will use a procedure called $CreateEmptyTail_L(x_0)$, which is defined as follows: Add an infinite chain of new empty worlds $(x_1, x_2, \ldots)$ to $W$ and set $R = R \cup \{(x_i, x_{i+1}) \mid i \geq 0\}$. (Note that the chain can be coded as a finite chain, which can be dynamically expanded when necessary).

**Algorithm 3.1.**
*Input:* A positive modal logic program $P$
in a modal logic $L \in \{KD, T, KDB, B\}$
*Output:* The least $L$-model $M = \langle W, \tau, R, h \rangle$ of $P$.
*Steps:*

1. Set $W = \{\tau\}$, $H(\tau) = P$, $R = \emptyset$.
   $CreateEmptyTail_L(\tau)$.
   Set $R = Ext_L(R)$.

2. For every world $u \in W$ with $H(u)$ not empty, and for every formula $\phi \in H(u)$:

   (a) Case $\phi = (A \leftarrow B_1, \ldots, B_k)$, where $k \geq 1$, and $M, u \vDash B_i$ for all $1 \leq i \leq k$: Set $H(u) = H(u) \cup \{A\}$.

   (b) Case $\phi = \Box\psi$: For every world $v \in W$ satisfying $R(u, v)$, set $H(v) = H(v) \cup \{\psi\}$.

   (c) Case $\phi = \Diamond\psi$, and $\neg(\exists x\, R(u, x) \wedge \psi \in H(x))$:

   Let $v$ be a new world.
   Set $W = W \cup \{v\}$, $H(v) = \{\psi\}$, $R = R \cup \{(u, v)\}$.
   $CreateEmptyTail_L(v)$.
   Set $R = Ext_L(R)$.

3. While some change occurred, repeat step 2.

Figure 2 illustrates the execution of this algorithm for the program $P = \{p, \Box s, [\Diamond q \leftarrow p], \Box[t \leftarrow q, s]\}$ in the logic *KD*.

The above algorithm always terminates. During its execution, at any enumerated step, the model $M$ is always less than or equal to any $L$-model of $P$. The following theorem is the main result of this subsection.

**Theorem 3.2.** For any positive modal logic program $P$ of size $n$, the model $M$ constructed by Algorithm 3.1 in a modal logic $L \in \{KD, T, KDB, B\}$ is the least $L$-model of $P$. If the modal depth of $P$ is finitely bounded, then $M$ can be constructed in PTIME, and for any $m \geq 1$, the size of the model $M|_m$ is bounded by a polynomial in $n$ and $m$.

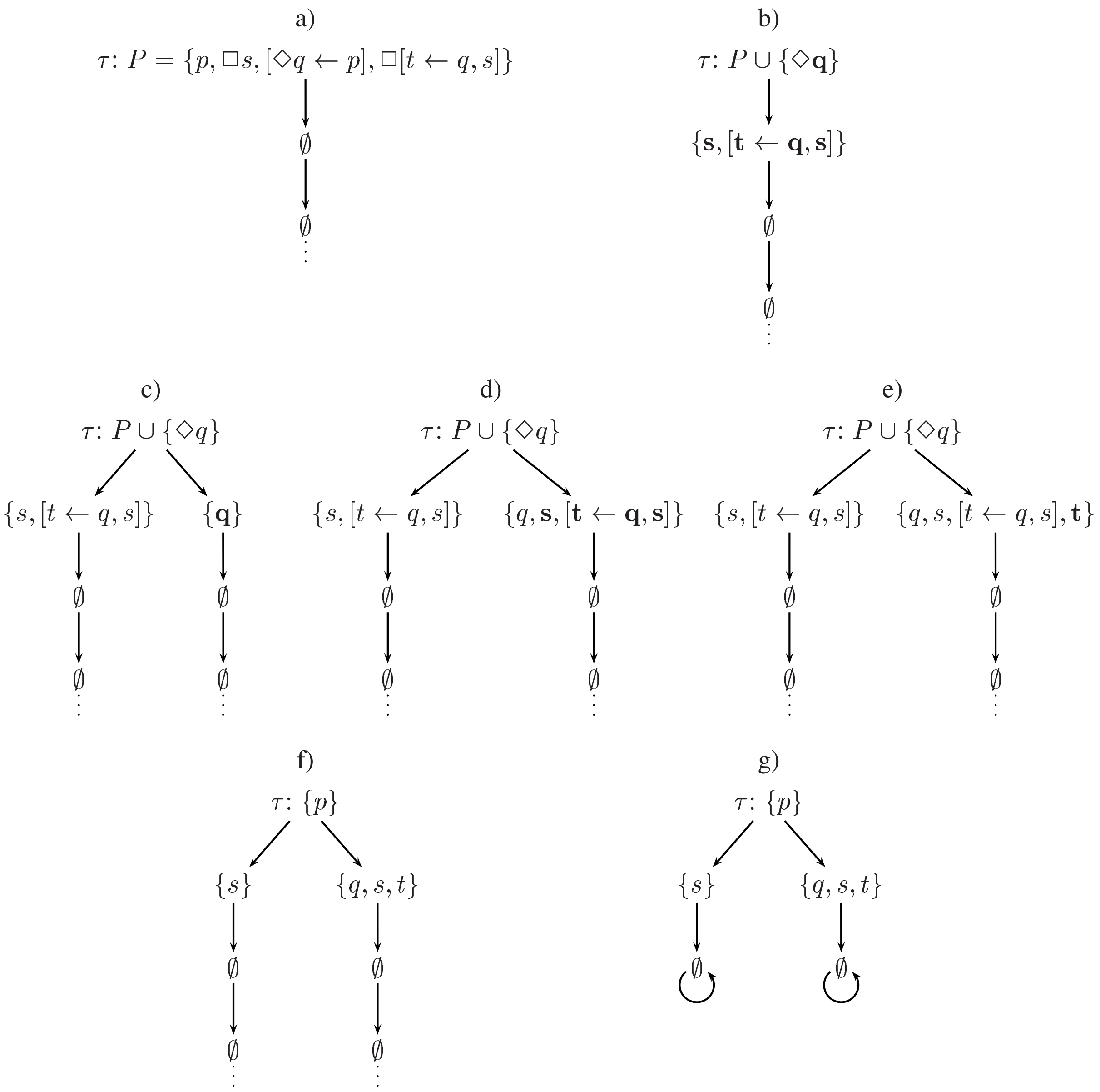


Figure 2. From a) to e): model graphs built during the execution of Algorithm 3.1 for the program $P = \{p, \Box s, [\Diamond q \leftarrow p], \Box[t \leftarrow q, s]\}$ in the logic *KD*. The model graph in a) is obtained after the execution of the step 1, the one in e) is the resulting model graph. The formulae in bold font indicate the differences w.r.t. the previous model graph. The model in f) corresponds to the model graph in e) and is the least *KD*-model of $P$. The model in g) is a simplified form of the one in f).

Although the model $M$ constructed by Algorithm 3.1 for a program $P$ is infinite, the essential part of $M$ is finite. For $L$ being $KD$ or $T$, $M$ can actually be transformed to an equivalent finite model. In that case, replacing all infinite chains of empty worlds in $M$ by a new empty world connected to itself, we can obtain the least $L$-model of $P$. For example, the model in Figure 1b is transformed to the one in Figure 1c, the model in Figure 2f is transformed to the one in Figure 2g. Thus, for any positive program $P$ with a finitely bounded modal depth, the least $KD$-model and the least $T$-model of $P$ can be constructed in PTIME, and their sizes are bounded by a polynomial in the size of $P$.

#### 3.2.2. The Case for the Logics *KD*5, *KD*45, and *S*5

Algorithm 3.1 can be used to construct the least $L$-model for a positive program in a logic $L \in \{KD5, KD45, S5\}$, provided that the procedure $CreateEmptyTail_L(x)$ now stands for "If $x = \tau$, then add two new empty worlds $\rho$ and $\varpi$ to $W$, and add the edges $(\tau, \rho)$ and $(\rho, \varpi)$ to $R$". This means that the infinite chains of new empty worlds are replaced by the two worlds $\rho$ and $\varpi$.

The following theorem concerns correctness and the complexity of Algorithm 3.1 for the logics *KD*5, *KD*45, and *S*5.

**Theorem 3.3.** For any positive modal logic program $P$ of size $n$, the model $M$ constructed by Algorithm 3.1 in a modal logic $L \in \{KD5, KD45, S5\}$ is the least $L$-model of $P$, it can be constructed in $O(n^4)$ steps and has size of order $O(n^4)$.

### 3.3. Characterizations of Positive Programs in *KB*, *K*5, *K*45, and *KB*5

For any positive program $P$ in a modal logic $L \in \{KB, K5, K45, KB5\}$ we can construct one or two $L$-models such that a positive formula $\phi$ follows from $P$ in $L$ iff $\phi$ is satisfied in these models.

**Definition 3.3. (Flat Models)**
We call a model $\langle W, \tau, R, h\rangle$ *flat* if $W = \{\tau\}$ and $R = \emptyset$.

**Definition 3.4. (The Least Flat Model of a Positive Program)**
A model $M$ is called the *least flat model* of a positive program $P$ if it is a flat model of $P$ and is less than or equal to any flat model of $P$.

**Definition 3.5. (Almost Serial Modal Logics)**
Let $L$ be a normal modal logic. We say that $L$ is *almost serial* if every $L$-frame $\langle W, \tau, R\rangle$ satisfies the formula $\exists x\ R(\tau, x) \to (\forall y\ R^*(\tau, y) \to \exists z\ R(y, z))$.

The logics $KB$, $K5$, $K45$, and $KB5$ are almost serial. Hence any non-flat connected frame in $KB$ (resp. $K5$, $K45$, $KB5$) is a frame in $KDB$ (resp. $KD5$, $KD45$, $S5$). We say that $KDB$, $KD5$, $KD45$, and $S5$ are the serial logics corresponding to $KB$, $K5$, $K45$, and $KB5$, respectively. For $L \in \{KB, K5, K45, KB5\}$, we use $LD$ to denote the serial logic corresponding to $L$.

**Theorem 3.4.** Let $P$ be a positive program in a modal logic $L \in \{KB, K5, K45, KB5\}$. Let $M$ be the least LD-model of $P$ constructed by Algorithm 3.1. If $P$ has no flat models then $M$ is the least $L$-model of $P$. If $P$ has a flat model then it has the least flat model $M'$, which can be constructed in PTIME and has size bounded by a polynomial in the size of $P$. The problem of checking whether a positive modal logic program $P$ has a flat model is decidable in PTIME. For every model $N$ of $P$, if $N$ is flat then $M' \leq N$, otherwise $M \leq N$.

Note that if $M$ is the least $L$-model of $P$ and $M'$ is the least flat model of $P$, then for every positive formula $\phi$, $P \vDash_L \phi$ iff $\phi$ is satisfied in both $M$ and $M'$.

By the above theorem, every positive program $P$ can be characterized in the logics $KB$, $K5$, $K45$, and $KB5$ by one or two models, which can be effectively constructed.

### 3.4. Constructing the Least Models for Positive Programs in *KD*4 and *S*4

Algorithm 3.1 cannot be applied for the logics $KD4$ and $S4$, because it will not terminate, for example, when $P$ contains the formula $\Box\Diamond p$. Before giving an algorithm of constructing the least models for positive programs in $KD4$ and $S4$, let us introduce some notations.

In Algorithm 3.5 given below, we use the following data structures:

- $M = \langle W, \tau, R^*, H \rangle$ - a model graph, with $R^*$ being the transitive closure of $R$. We sometimes refer to $M$ as a model in the same way as described in Section 3.2.
- $StatusW : W \to \{normal, minimal\}$;
- $StatusR : R \to \{permanent, temporal\}$;
- $Next : W \times \mathcal{F} \to W$, which is interpreted as follows: $Next(u, \Diamond\phi) = v$ means $R(u, v)$, $\phi \in H(v)$, and the world $u$ *realizes* the requirement $\Diamond\phi$ by going to $v$.

We define the binary relation $Permanent$ on $W$ as follows

$$Permanent(x, y) \equiv (R(x, y) \wedge StatusR(x, y) = permanent),$$

and use $Permanent^*$ to denote its transitive closure. We also need the following notation:

$$\begin{array}{rcl} H^*(x) & = & \{\phi \mid \exists y\ Permanent^*(y, x) \vee y = x \text{ and} \\ & & (\Box\phi \in H(y) \text{ or } (\phi \in H(y) \text{ and } \phi = \Box\psi \text{ for some } \psi))\}, \end{array}$$

where $H^*(x)$ can be interpreted as the potentiality inherited from both $x$ and its predecessors.

Consider Algorithm 3.5 given below. We build the least $L$-model for $P$ by constructing a $L$-model graph for it. The relation $Permanent$, without self-referencing edges, forms a tree with root $\tau$, which is a skeleton of the model graph.

Consider step 2c. If we create a new world to realize the formula $\Diamond p$, then it should contain $p$ and $H^*(u)$. We first try to find a world $v$ on the path from $\tau$ to $u$ (in the tree generated by $Permanent$) such that it contains $\Diamond p$, and $H^*(v) = H^*(u)$. If such $v$ exists and the formula $\Diamond p$ has been realized in $v$ by a connection from $v$ to $w$, then we just connect $u$ to $w$; but this edge is not $permanent$, because if a new formula $\Box\zeta$ is later added to $u$ while $\zeta$ or $\Box\zeta$ is not present in $w$ then we will delete the edge. If such $v$ exists but the formula $\Diamond p$ has not been realized in $v$, then we do nothing; this means that the formula $\Diamond p$ will be realized in $v$ before in $u$. If such $v$ does not exist, then we realize the formula $\Diamond p$ in $u$ by creating a new world, denoted by $v$, assigning $p$ and $H^*(u)$ to it, and connecting $u$ to $v$, and $v$ to a new $minimal$ world. $Minimal$ worlds are elements that make $M$ the least $L$-model of $P$. They have the same role as the infinite chains of empty worlds used in Algorithm 3.1. In step 2(b)i, we do not add $\psi$ to $u$ because of the nature of $minimal$ worlds.

In Algorithm 3.5, we use a procedure called $CreateNextMinNode(x)$, which is defined as:

Let $y$ be a new world.
Set $W = W \cup \{y\}$, $StatusW(y) = minimal$, $H(y) = H^*(x)$,
$R = R \cup \{(x, y), (y, y)\}$, $StatusR(x, y) = permanent$.
If $L = S4$, set $StatusR(y, y) = permanent$;
otherwise, set $StatusR(y, y) = temporal$.

**Algorithm 3.5.**
*Input:* A positive modal logic program $P$ in a modal logic $L \in \{KD4, S4\}$
*Output:* The least $L$-model $M = \langle W, \tau, R^*, h\rangle$ of $P$.
*Steps:*

1. Set $W = \{\tau\}$, $StatusW(\tau) = normal$, $H(\tau) = P$.
   If $L = S4$, set $R = \{(\tau, \tau)\}$ and $StatusR(\tau, \tau) = permanent$;
   otherwise, set $R = \emptyset$.
   $CreateNextMinNode(\tau)$.
2. For every world $u \in W$, and for every formula $\phi \in H(u)$,
   (a) Case $\phi = B \leftarrow C_1, \dots, C_k$ for some $k \geq 1$, and $M, u \models C_i$ for all $1 \leq i \leq k$:
      Set $H(u) = H(u) \cup \{B\}$.
   (b) Case $\phi = \Box\psi$ for some $\psi$ :
      i. If $StatusW(u) = minimal$, $StatusR(u, u) = temporal$, and $\psi \notin H(u)$, then
         Set $R = R - \{(u, u)\}$, $StatusW(u) = normal$,
         $CreateNextMinNode(u)$.
      ii. Otherwise,
         For every $v$ such that $Permanent^*(u, v)$,
            Set $H(v) = H(v) \cup \{\phi, \psi\}$,
         For every $v$ such that $R(u, v)$, $StatusR(u, v) = temporal$,
         and $\{\phi, \psi\} \nsubseteq H(v)$,
            Set $R = R - \{(u, v)\}$,
            For every $\Diamond p$ such that $Next(u, \Diamond p) = v$,
               Set $Next(u, \Diamond p)$ undefined.
   (c) Case $\phi = \Diamond p$ for some primitive proposition $p$, and $Next(u, \Diamond p)$ is undefined:
      i. If there exist $v$ and $w$ such that:
         $$Permanent^*(v, u) \wedge H^*(v) = H^*(u) \wedge \phi \in H(v) \wedge$$
         $$Permanent(v, w) \wedge Next(v, \Diamond p) = w,$$
         Set $R = R \cup \{(u, w)\}$, $StatusR(u, w) = temporal$, $Next(u, \Diamond p) = w$.
      ii. Otherwise, if $\neg(\exists x \;\; x \neq u \wedge Permanent^*(x, u) \wedge H^*(x) = H^*(u) \wedge \phi \in H(x))$,

Let $v$ be a new world, and set $Next(u, \Diamond p) = v$.
Set $W = W \cup \{v\}$, $StatusW(v) = normal$, $H(v) = H^*(u) \cup \{p\}$,
$R = R \cup \{(u, v)\}$, $StatusR(u, v) = permanent$.
If $L = S4$, set $R = R \cup \{(v, v)\}$, $StatusR(v, v) = permanent$.
$CreateNextMinNode(v)$.

3. While some change occurred, repeat step 2.

It is shown in [9] that this algorithm always terminates and during its execution, at any enumerated step, the model $M$ is always less than or equal to any $L$-model of $P$. The following theorem is the main result of this subsection.

**Theorem 3.6.** For any positive program $P$ in a modal logic $L \in \{KD4, S4\}$, the resulting model $M$ of the execution of Algorithm 3.5 for $P$ is the least $L$-model of $P$. Moreover, if the modal depth of $P$ and the number of rules of the form $\Box^s(A \leftarrow B_1, \ldots, B_p)$, where $s \geq 1$, $p \geq 1$, and $A$ is of the form $\Box p$ or $\Diamond p$, are finitely bounded, then $M$ can be constructed in PTIME and its size is bounded by a polynomial in the size of $P$.

Note that the modal depth of any program in $S4$ can be assumed to be less than or equal to 2, since in $S4$ we have $\Box\Box\phi \equiv \Box\phi$.

# 4. Fixed-Domain First-Order Modal Logics over Signatures without Functions

In the previous sections we have dealt with positive *propositional* modal logic programs. A real query language, however, should allow to use *universal first-order* rules. Such a query language for modal logics, called MDatalog, is proposed in the next section. In this section, we recall the definition of fixed-domain first-order modal logics over signatures without functions. MDatalog programs are interpreted in such logics.

Our language now contains additionally a signature without functions $\Sigma$, a countable set of variables, and the quantifiers $\forall$ and $\exists$.

A *term* is either a constant symbol or a variable. We call $R_i(t_1, \ldots, t_n)$, where $R_i$ is a relation symbol of arity $n$ and $t_1, \ldots, t_n$ are terms, a *classical atom*.

A *first-order modal formula*, hereafter simply called a formula if there is no ambiguity, is any finite sequence of symbols of our language obtained by applying the following rules: any classical atom is a formula, and if $\phi$ and $\psi$ are formulae, and $x$ is a variable, then $\neg\phi$, $\phi \vee \psi$, $\phi \wedge \psi$, $\phi \to \psi$, $\Box\phi$, $\Diamond\phi$, $\forall x\ \phi$, and $\exists x\ \phi$ are formulae.

A formula is called *ground* if it contains no quantifiers and no variables. *Herbrand base* is the set of all ground classical atoms. We use letters $a$, $b$, $c$ to denote constant symbols.

**Definition 4.1. (First-Order Kripke Models)**
A *fixed-domain first-order Kripke model over a signature without functions* $\Sigma$, hereafter simply called a first-order Kripke model or just a Kripke model if there is no ambiguity, is a tuple $M = \langle D, W, \tau, R, m \rangle$, where $D$ is a set called *domain*, $\langle W, \tau, R \rangle$ is a Kripke frame, and $m$ is an interpretation of constant symbols and relation symbols. For a constant symbol $c_i$, $m(c_i)$ is an element of $D$, denoted also by $c_i^M$.

For a relation symbol $R_j$ of arity $n$ and a world $w \in W$, $m(w)(R_j)$ is a $n$-ary relation on $D$, denoted also by $R_j^{M,w}$.

**Definition 4.2. (Variable Assignments)**
A *variable assignment* on a Kripke model $M$ is a function $V$ that maps each variable $x$ to an element of the domain of $M$. The value of a term $t$ wrt. a variable assignment $V$ is denoted by $V(t)$ and defined as follows: If $t$ is a constant symbol $c_i$ then $V(t) = c_i^M$; if $t$ is a variable $x$ then $V(t) = V(x)$.

**Definition 4.3. (Satisfiability)**
The *satisfiability relation* $M, V, w \vDash \phi$, where $M = \langle D, W, \tau, R, m\rangle$ is a Kripke model, $V$ is a variable assignment on $M$, $w \in W$, and $\phi$ is a formula, is recursively defined similarly as for the propositional case except that

$$\begin{array}{lll} M, V, w \vDash R_j(t_1, \ldots, t_n) & \text{iff} & R_j^{M,w}(V(t_1), \ldots, V(t_n)); \\ M, V, w \vDash \forall x\, \phi(x) & \text{iff} & \text{for all } a \in D, (M, V', w \vDash \phi(a)), \\ & & \text{where } V'(x) = a \text{ and } V'(y) = V(y) \text{ for } y \neq x; \\ M, V, w \vDash \exists x\, \phi(x) & \text{iff} & \text{there exists } a \in D \text{ such that } M, V', w \vDash \phi(a), \\ & & \text{where } V'(x) = a \text{ and } V'(y) = V(y) \text{ for } y \neq x. \end{array}$$

If $M, V, w \vDash \phi$ then we say that $\phi$ is satisfied at $w$ in $M$ wrt. $V$. We say that $\phi$ is satisfied in $M$ wrt. $V$, and write $M, V \vDash \phi$, if $M, V, \tau \vDash \phi$. We say that $M$ satisfies $\phi$, or $\phi$ is satisfied in $M$, and write $M \vDash \phi$, if for any variable assignment $V$, $M, V \vDash \phi$. If $\phi$ is a ground formula then we write $M, w \vDash \phi$ to denote that $M, V, w \vDash \phi$ for some $V$ (note that what $V$ does not matter).

Let $L$ be the name of some propositional normal modal logic given in Table 2, e.g. $KD$. To define what is the *first-order modal logic* $L$ we define its class of admissible interpretations to be the class of first-order Kripke models $M = \langle D, W, \tau, R, m\rangle$ with $\langle W, \tau, R\rangle$ being a $L$-frame.

A number of definitions for propositional modal logics can be shifted in a straightforward way to the first-order version, including the definitions of modal depth of a formula, $L$-satisfiability, and $L$-model.

The *quantifier depth* of a formula $\phi$ is defined to be the maximal nesting depth of quantifiers occurring in $\phi$.

## 5. The Modal Query Language MDatalog

In this section we extend the definition of positive propositional modal logic programs to formulate first-order ones, called MDatalog programs. We also define what is a query to an MDatalog program.

**Definition 5.1. (MDatalog Programs)**
A *MDatalog program* is a finite set of rules of the form

$$\forall \overline{x}\, \Box^s [A \leftarrow B_1, \ldots, B_k]$$

where $s \geq 0$, $k \geq 0$, and

1. $A$, $B_1, \ldots, B_k$ are formulae of the form $\Box\phi$, $\Diamond\phi$, or $\phi$, with $\phi$ being a classical atom.
2. $\overline{x}$ are all variables occurring in $(A \leftarrow B_1, \ldots, B_k)$.

3. If $k = 0$ then the rule is a ground fact, i.e. there is no quantifiers prefixing the rule.

4. All variables of $A$ also occur in $(B_1, \ldots, B_k)$.

**Definition 5.2. (Queries)**
A *query* is a formula of the form $\exists \overline{x}\ \phi(\overline{x})$, where $\phi$ is a positive formula without quantifiers, and $\overline{x}$ is a vector of all variables of $\phi$, and can be empty.

Fix a modal logic $L$. Given an MDatalog program $P$ and a query $\exists \overline{x}\ \phi(\overline{x})$, the goal is to check whether $P \models_L \exists \overline{x}\ \phi(\overline{x})$, and to find tuples $\overline{c}$ of constant symbols such that $P \models_L \phi(\overline{c})$.

In the remainder of this section we show that the language MDatalog has the same expressiveness as its extended version defined below.

**Definition 5.3. (Non-negative Modal Horn Formulae)**
A formula $\phi$ is called a *non-negative modal Horn formula* iff one of the following conditions holds:

- $\phi$ is a classical atom;
- $\phi$ is of the form $\psi \leftarrow \zeta$, where $\psi$ is a non-negative modal Horn formula and $\zeta$ a formula constructed from classical atoms and the connectives[1] $\wedge$, $\Box$, $\Diamond$;
- $\phi = \Box\psi$, or $\phi = \Diamond\psi$, or $\phi = \psi \wedge \zeta$, where $\psi$ and $\zeta$ are non-negative modal Horn formulae.

If we define a program rule to be a universally quantified non-negative modal Horn formula, then what should correspond to the conditions 3 and 4 of Definition 5.1? To deal with this question let us define a constraint $VarConstr(\phi, V)$, for $\phi$ being a non-negative modal Horn formula and $V$ a set of variables, as follows:

$$\begin{aligned}
VarConstr(A, V) &\equiv Var(A) \subseteq V \\
VarConstr(\psi \leftarrow \zeta, V) &\equiv VarConstr(\psi, Var(\zeta) \cup V) \\
VarConstr(\psi \wedge \zeta, V) &\equiv VarConstr(\psi, V) \wedge VarConstr(\zeta, V) \\
VarConstr(\Box\psi, V) &\equiv VarConstr(\psi, V) \\
VarConstr(\Diamond\psi, V) &\equiv VarConstr(\psi, V)
\end{aligned}$$

where $A$ is a classical atom, and $Var(\phi)$ denotes the set of variables of $\phi$.

We now define a language called eMDatalog, which is an extended version of MDatalog.

**Definition 5.4. (eMDatalog Programs)**
An *eMDatalog program* is a finite set of formulae of the form $\forall \overline{x}\ \phi$, where $\phi$ is a non-negative modal Horn formula, $\overline{x}$ are all variables of $\phi$, and the constraint $VarConstr(\phi, \emptyset)$ is true.

As stated by the following proposition, MDatalog has the same expressiveness as eMDatalog.

**Proposition 5.1.** For any eMDatalog program $P$, there exists an MDatalog program $P'$ such that for any query $\exists \overline{x}\ \phi(\overline{x})$, the answers wrt. $P'$ are exactly the answers wrt. $P$. Moreover, $P'$ can be obtained from $P$ in a polynomial time in the size of $P$, the size of $P'$ is bounded by a polynomial in the size of $P$, the modal depth of $P'$ is equal to the modal depth of $P$, and the quantifier depth of $P'$ is not greater than the quantifier depth of $P$.

---
[1]The connective $\vee$ is excluded in order to deal with the conditions 3 and 4 of Definition 5.1.

**Proof:**
Given an eMDatalog program $P$, we translate it to an MDatalog program by applying Mints translation. 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 $A$ denotes a formula of the form $\Box\phi$, $\Diamond\phi$, or $\phi$, where $\phi$ is a classical atom. The symbols $X$, $Y$ denote sequences of formulae. The relational symbols of the form $P_\phi$ or $P_X$ are new (wrt. the primary signature), and when we write $P_\phi(\overline{y})$ (resp. $P_X(\overline{y})$) then $\overline{y}$ are all variables of $\phi$ (resp. of $X$). In the rule (2), $\overline{y}$ are all variables of $\phi$, and $\overline{z}$ all variables of $\psi$. In the rules (1), (3), (4), (7), and (8), we require $\phi$ not to be a classical atom. We apply the translation rules to $P$ until no further changes can be made. Let $P'$ denote the resulting set.

$$
\begin{array}{rll}
\forall\overline{x}\,\Box^k\Diamond\phi & \forall\overline{x}\,\Box^k\Diamond P_\phi(\overline{x}),\ \forall\overline{x}\,\Box^{k+1}[\phi \leftarrow P_\phi(\overline{x})] & (1)\\
\forall\overline{x}\,\Box^k(\phi\wedge\psi) & \forall\overline{y}\,\Box^k\phi,\ \forall\overline{z}\,\Box^k\psi & (2)\\
\forall\overline{x}\,\Box^k[\Box\phi \leftarrow X] & \forall\overline{x}\,\Box^k[\Box P_\phi(\overline{y}) \leftarrow X],\ \forall\overline{y}\,\Box^{k+1}[\phi \leftarrow P_\phi(\overline{y})] & (3)\\
\forall\overline{x}\,\Box^k[\Diamond\phi \leftarrow X] & \forall\overline{x}\,\Box^k[\Diamond P_\phi(\overline{y}) \leftarrow X],\ \forall\overline{y}\,\Box^{k+1}[\phi \leftarrow P_\phi(\overline{y})] & (4)\\
\forall\overline{x}\,\Box^k[(\phi\wedge\psi) \leftarrow X] & \forall\overline{x}\,\Box^k[\phi \leftarrow P_X(\overline{y})],\ \forall\overline{x}\,\Box^k[\psi \leftarrow P_X(\overline{y})],\ \forall\overline{y}\,\Box^k[P_X(\overline{y}) \leftarrow X] & (5)\\
\forall\overline{x}\,\Box^k[(\phi\leftarrow\psi) \leftarrow X] & \forall\overline{x}\,\Box^k[\phi \leftarrow \psi, X] & (6)\\
\forall\overline{x}\,\Box^k[A \leftarrow X, \Box\phi, Y] & \forall\overline{x}\,\Box^k[A \leftarrow X, \Box P_\phi(\overline{y}), Y],\ \forall\overline{y}\,\Box^{k+1}[P_\phi(\overline{y}) \leftarrow \phi] & (7)\\
\forall\overline{x}\,\Box^k[A \leftarrow X, \Diamond\phi, Y] & \forall\overline{x}\,\Box^k[A \leftarrow X, \Diamond P_\phi(\overline{y}), Y],\ \forall\overline{y}\,\Box^{k+1}[P_\phi(\overline{y}) \leftarrow \phi] & (8)\\
\forall\overline{x}\,\Box^k[A \leftarrow X, \phi\wedge\psi, Y] & \forall\overline{x}\,\Box^k[A \leftarrow X, \phi, \psi, Y]. & (9)
\end{array}
$$

It is easily seen that $P'$ is an MDatalog program, the modal depth of $P'$ is equal to the modal depth of $P$, the quantifier depth of $P'$ is not greater than the quantifier depth of $P$, the size of $P'$ is of order $O(n^2)$, and the translation is done in $O(n^2)$ steps, where $n$ is the size of $P$.

Each model of $P'$ is also a model of $P$. Each model $M$ of $P$ can be extended to a model $M'$ of $P'$ in the way such that when restricted to the primary signature, $M'$ is equal to $M$. Hence, for any query $\exists\overline{x}\ \phi(\overline{x})$ (in the primary signature), the answers wrt. $P'$ are exactly the answers wrt. $P$. $\Box$

# 6. Ordering First-Order Kripke Models

We can order first-order Kripke models similarly as we do for propositional Kripke models. The conditions about frame, i.e. the first three ones in the following definition, remain unchanged.

**Definition 6.1. (An Order between First-Order Kripke Models)**
Let $M = \langle D, W, \tau, R, m\rangle$ and $N = \langle D', W', \tau', R', m'\rangle$ be Kripke models. We say that $M$ is less than or equal to $N$ with respect to a function $f : D \to D'$ and a binary relation $r \subseteq W \times W'$ if

1. $r(\tau, \tau')$
2. $\forall x, x', y\ \ R(x, y) \wedge r(x, x') \to \exists y'\ R'(x', y') \wedge r(y, y')$
3. $\forall x, x', y'\ \ R'(x', y') \wedge r(x, x') \to \exists y\ R(x, y) \wedge r(y, y')$
4. For any constant symbol $c_i$, $f(c_i^M) = c_i^N$.

5. For any $x \in W$ and $x' \in W'$ such that $r(x, x')$, for any $n$-ary relation symbol $R_j$ and elements $a_1$, …, $a_n$ of $D$, if $R_j^{M,x}(a_1, \ldots, a_n)$ then $R_j^{N,x'}(f(a_1), \ldots, f(a_n))$.

We say that a model $M$ is less than or equal to $N$, and write $M \leq N$, if $M$ is less than or equal to $N$ with respect to some $f$ and some $r$.

In the above definition, $r(x, x')$ intuitively states that the world $x$ is less than or equal to $x'$.

**Lemma 6.1.** Let $M = \langle D, W, \tau, R, m\rangle$ and $N = \langle D', W', \tau', R', m'\rangle$ be Kripke models. Let $M \leq N$ wrt. $f$ and $r$. Then for any $u \in W$ and $u' \in W'$ such that $r(u, u')$, for any positive ground formula $\phi$, if $M, u \vDash \phi$ then $N, u' \vDash \phi$.

**Proof:**
We prove the lemma by induction on the construction of $\phi$. Suppose that $r(u, u')$ and $M, u \vDash \phi$ hold.

Case $\phi = R_j(c_{k_1}, \ldots, c_{k_n})$ : By definition, $R_j^{M,u}(c_{k_1}^M, \ldots, c_{k_n}^M)$ holds. Since $r(u, u')$ holds, it follows that $R_j^{N,u'}(f(c_{k_1}^M), \ldots, f(c_{k_n}^M))$. Hence $R_j^{N,u'}(c_{k_1}^N, \ldots, c_{k_n}^N)$ holds, and finally $N, u' \vDash R_j(c_{k_1}, \ldots, c_{k_n})$.

The case $\phi = \psi \wedge \zeta$ or $\phi = \psi \vee \zeta$ is trivial.

Case $\phi = \Box\psi$ : By the condition 3, we have $\forall v' \;\; R'(u', v') \rightarrow \exists v \; r(v, v') \wedge R(u, v) \wedge (M, v \vDash \psi)$. By the inductive assumption, we have $\forall v' \;\; R'(u', v') \rightarrow (N, v' \vDash \psi)$, hence $N, u' \vDash \Box\psi$.

Case $\phi = \Diamond\psi$ : Since $M, u \vDash \Diamond\psi$, it follows that $\exists v \;\; R(u, v) \wedge (M, v \vDash \psi)$. By the condition 2, we have $\exists v, v' \;\; R'(u', v') \wedge r(v, v') \wedge (M, v \vDash \psi)$. By the inductive assumption, it follows that $\exists v' \;\; R'(u', v') \wedge (N, v' \vDash \psi)$. Therefore $N, u' \vDash \Diamond\psi$. $\square$

**Corollary 6.1.** If $M \leq N$ and $\phi$ is a positive ground formula satisfied in $M$, then $N \vDash \phi$.

It can be shown that the relation $\leq$ between first-order Kripke models is a pre-order.

## 7. Instantiating MDatalog Programs

Given an MDatalog program, one can translate it to a positive propositional modal logic program by generating all ground instances of its rules. Theorems about existence of the least models and complexity results for positive propositional modal logic programs can be then reformulated for MDatalog programs.

**Lemma 7.1.** Let $M_0 = \langle W, \tau, R, h\rangle$ be a propositional Kripke model over the language that takes Herbrand base as the set of primitive propositions. Let $M = \langle D, W, \tau, R, m\rangle$ be a first-order Kripke model such that for any $n$-ary relation symbol $R_j$ and constant symbols $c_{k_1}$, …, $c_{k_n}$, and for any world $w \in W$

$$R_j^{M,w}(c_{k_1}^M, \ldots, c_{k_n}^M) \equiv (R_j(c_{k_1}, \ldots, c_{k_n}) \in h(w)).$$

Then for any ground formula $\phi$, $M \vDash \phi$ iff $M_0 \vDash \phi$.

**Proof:**
We show that for any world $w \in W$ and for any ground formula $\phi$, $M, w \vDash \phi$ iff $M_0, w \vDash \phi$. It suffices

to show this for $\phi$ being a classical atom; other cases can be easily proved by induction. We have:

$$\begin{array}{ll} & M, w \vDash R_j(c_{k_1}, \ldots, c_{k_n}) \\ \text{iff} & R_j^{M,w}(c_{k_1}^M, \ldots, c_{k_n}^M) \\ \text{iff} & R_j(c_{k_1}, \ldots, c_{k_n}) \in h(w) \\ \text{iff} & M_0, w \vDash R_j(c_{k_1}, \ldots, c_{k_n}). \end{array}$$

□

**Corollary 7.1.** Let $M_0 = \langle W, \tau, R, h\rangle$ be a model such as in the preceding lemma. Let $M_1 = \langle Const(\Sigma), W, \tau, R, m\rangle$ be the first-order Kripke model such that $Const(\Sigma)$ is the set of all constant symbols of $\Sigma$, $c_i^{M_1} = c_i$ for any constant symbol $c_i$, and that for any $n$-ary relation symbol $R_j$ and constant symbols $c_{k_1}, \ldots, c_{k_n}$, for any world $w \in W$,

$$R_j^{M_1,w}(c_{k_1}, \ldots, c_{k_n}) \equiv (R_j(c_{k_1}, \ldots, c_{k_n}) \in h(w)).$$

Then for any ground formula $\phi$, $M_1 \vDash \phi$ iff $M_0 \vDash \phi$.

In fact, the models $M_0$ and $M_1$ in the above corollary are the same model, looked at different views. So, we can treat any propositional Kripke model over the language that takes Herbrand base as the set of primitive propositions as a first-order Kripke model.

**Definition 7.1. (The Ground Instantiation of an MDatalog Program)**
For an MDatalog program $P$, we define the *ground instantiation of* $P$, denoted by $P_0$, to be the set of all ground instances of all rules of $P$, substituted over the set of constant symbols occurring in $P$. We can treat $P_0$ as a positive propositional modal logic program that takes Herbrand base as the set of primitive propositions.

**Theorem 7.1.** Let $P$ be an MDatalog program and $P_0$ be its ground instantiation. Let $L$ be some normal modal logic. If $M_0$ is the least propositional $L$-model of $P_0$ then $M_0$ is the least first-order $L$-model of $P$.

**Proof:**
Let $M_0 = \langle W, \tau, R, h\rangle$, and let $M_1$ be the model defined as in Corollary 7.1. We show that $M_1$ is the least $L$-model of $P$. Let $N = \langle D', W', \tau', R', m'\rangle$ be an arbitrary first-order $L$-model of $P$. Let $N_0 = \langle W', \tau', R', h'\rangle$ be the propositional model such that for any $n$-ary relation symbol $R_j$ and constant symbols $c_{k_1}, \ldots, c_{k_n}$, for any world $w \in W$,

$$(R_j(c_{k_1}, \ldots, c_{k_n}) \in h'(w)) \equiv R_j^{N,w}(c_{k_1}^N, \ldots, c_{k_n}^N).$$

By Lemma 7.1, $N_0$ is a model of $P_0$. Hence there exists a relation $r \subseteq W \times W'$ such that $M_0 \leq N_0$ wrt. $r$. Let $f : Const(\Sigma) \to D'$ be the function defined by $f(c_i) = c_i^N$, for any constant symbol $c_i$. We show that $M_1 \leq N$ wrt. $f$ and $r$. It suffices to show that for any $x \in W$ and $x' \in W'$ such that $r(x, x')$, for any $n$-ary relation symbol $R_j$ and constant symbols $c_{k_1}, \ldots, c_{k_n}$, if $R_j^{M_1,x}(c_{k_1}, \ldots, c_{k_n})$ then $R_j^{N,x'}(f(c_{k_1}), \ldots, f(c_{k_n}))$. Suppose that $R_j^{M_1,x}(c_{k_1}, \ldots, c_{k_n})$ holds. By the definition of $M_1$, we have $R_j(c_{k_1}, \ldots, c_{k_n}) \in h(x)$. Since $M_0 \leq N_0$ wrt. $r$ and $r(x, x')$, it follows that $R_j(c_{k_1}, \ldots, c_{k_n}) \in h(x')$. By the definition of $N_0$, $R_j^{N,x'}(c_{k_1}^N, \ldots, c_{k_n}^N)$ holds. Hence $R_j^{N,x'}(f(c_{k_1}), \ldots, f(c_{k_n}))$ holds. This completes our proof. □

**Corollary 7.2.** Let $L$ be one of the modal logics $KD$, $T$, $KDB$, $B$, $KD5$, $KD45$, $S5$, $KD4$, and $S4$. Let $P$ be an MDatalog program. Then there exists the least $L$-model of $P$, which can be effectively constructed. Moreover, if the quantifier depth of $P$ is finitely bounded then

- If $L \in \{KD5, KD45, S5\}$, or $L \in \{KD, T\}$ and additionally the modal depth of $P$ is finitely bounded, then the least $L$-model of $P$ can be constructed in PTIME and its size is bounded by a polynomial in the size of $P$.
- If $L$ is $KDB$ or $B$, and the modal depth of $P$ is finitely bounded, then the least $L$-model $M$ of $P$ can be constructed in PTIME, and for any $m \geq 1$ the size of the model $M|_m$ is bounded by a polynomial in the size of $P$ and $m$.

**Proof:**
Let $P_0$ be the ground instantiation of $P$. Just note that if the quantifier depth of $P$ is finitely bounded then the size of $P_0$ is bounded by a polynomial in the size of $P$, and the modal depth of $P_0$ is equal to the modal depth of $P$. This corollary then follows from Theorems 3.2, 3.3, 3.6, and 7.1. □

The definitions of *flat models* and the *least flat model* of an MDatalog program are specified in the same way as for the propositional case.

**Theorem 7.2.** Let $P$ be an MDatalog program and $P_0$ be its ground instantiation. If $M_0$ is the least flat model of $P_0$ then it is also the least flat model of $P$.

This theorem can be proved in the same way as Theorem 7.1. As a consequence, we have the following corollary for the almost serial modal logics $KB$, $K5$, $K45$, and $KB5$.

**Corollary 7.3.** Let $L$ be one of the modal logics $KB$, $K5$, $K45$, and $KB5$. Let $P$ be an MDatalog program. Let $M$ be the least $LD$-model of $P$. If $P$ has no flat models then $M$ is the least $L$-model of $P$. If $P$ has a flat model then it has the least flat model $M'$, and for every model $N$ of $P$, if $N$ is flat then $M' \leq N$, otherwise $M \leq N$. Moreover, if the quantifier depth of $P$ is finitely bounded then

- The model $M'$, if exists, can be constructed in PTIME and its size is bounded by a polynomial in the size of $P$.
- If $L$ is one of the logics $K5$, $K45$, and $KB5$, then the model $M$ can be constructed in PTIME and its size is bounded by a polynomial in the size of $P$.
- If $L$ is $KB$ and the modal depth of $P$ is finitely bounded then the model $M$ can be constructed in PTIME, and for any $m \geq 1$, the size of $M|_m$ is bounded by a polynomial in the size of $P$ and $m$.

For the same reasons as for Corollary 7.2, this corollary follows from Theorems 3.2, 3.3, 3.4, 7.1, and 7.2.

The following theorem concerns the complexity of MDatalog language.

**Theorem 7.3.** Let $L$ be one of the modal logics $KD$, $T$, $KB$, $KDB$, $B$, $K5$, $KD5$, $K45$, $KD45$, $KB5$, and $S5$. Given an MDatalog program $P$ and a query $\exists \overline{x}\ \phi(\overline{x})$ such that the quantifier depths of $P$ and the query are finitely bounded, and that the modal depth of $P$ is finitely bounded in the case when $L$ is not an extension of $K5$, then the problem of checking whether $P \vDash_L \exists \overline{x}\ \phi(\overline{x})$ and finding all tuples $\overline{c}$ of constant symbols such that $P \vDash_L \phi(\overline{c})$ is solvable in PTIME.

**Proof:**
If $L$ is a serial modal logic then let $M$ be the least $L$-model of $P$. If $L$ is an almost serial but not serial modal logic then let $M$ be the least $LD$-model of $P$, and if $P$ has a flat model[2], then let $M'$ be the least flat model of $P$. Let $m$ be the modal depth of $\phi(\overline{x})$. Under the assumptions of the theorem, both $M$ and $M'$ can be constructed in PTIME, the size of $M'$ is bounded by a polynomial in the size of $P$, and the size of $M|_m$ is bounded by a polynomial in the size of $P$ and $m$.

Observe that $P \vDash_L \exists\overline{x}\ \phi(\overline{x})$ iff there exists a tuple $\overline{c}$ of constant symbols occurring in $P$ such that $P \vDash_L \phi(\overline{c})$. The number of such tuples is bounded by a polynomial in the size of $P$.

If $L$ is a serial modal logic or $P$ has no flat models, then for any tuple $\overline{c}$ of constant symbols, $P \vDash_L \phi(\overline{c})$ iff $M|_m \vDash \phi(\overline{c})$. If $L$ is an almost serial but not serial modal logic and $P$ has a flat model, then for any tuple $\overline{c}$ of constant symbols, $P \vDash_L \phi(\overline{c})$ iff $\phi(\overline{c})$ is satisfied in both $M|_m$ and $M'$. Thus, for both of these cases, the problem of checking whether $P \vDash_L \phi(\overline{c})$ is solvable in polynomial time. Therefore the theorem is true. □

## 8. Discussion on Implementation

In this section we discuss how algorithms of constructing the least models for MDatalog programs can be implemented, and describe a technique that optimizes both execution time and memory for the algorithms. Claims of this section are given without proof.

### 8.1. Algorithms of Constructing the Least Models

Given an MDatalog program $P$, we will rather not generate the ground instantiation of $P$ instantly, because doing so would generate a large program that could contain many irrelevant rules. We can apply traditional methods that have been used for positive classical logic programs (i.e. Datalog programs) to avoid generating instances of rules.

Recall that for Algorithms 3.1 and 3.5 that construct the least propositional Kripke models for positive propositional modal logic programs, in general we have to deal with three cases when the considered formula is of the form: a) $(A \leftarrow B_1, \ldots, B_k)$, b) $\Box\phi$, or c) $\Diamond\phi$. When trying to rewrite those algorithms for MDatalog programs, if we convert all of the rules to the equivalent form $\Box^s\ \forall\overline{x}\ (A \leftarrow B_1, \ldots, B_k)$ then the steps for the cases b) and c) need not to be changed; we only have to rewrite the step numbered 2a) in those algorithms to solve the case when the considered formula is of the form $\forall\overline{x}\ (A \leftarrow B_1, \ldots, B_k)$, with $k \geq 1$.

To keep classical atoms in memory we use new relation symbols $R_1{}', R_2{}', \ldots$ in the following way: if an atom $R_j(c_{k_1}, \ldots, c_{k_n})$ is present at a world $w$ then we record this fact by keeping $R_j{}'(w, c_{k_1}, \ldots, c_{k_n})$. So, if we pay attention only to classical atoms then we have a classical relational database. Denote this database by $RDB$. Let $RDB$ contain also the accessibility relation.

Suppose that it is time we want to realize some rule $\forall\overline{x}\ (A \leftarrow B_1, \ldots, B_k)$, where $k \geq 1$, at some world $u$. We will compute tuples $\overline{c}$ of constants that in the place of $\overline{x}$ satisfy the set $\{B_1, \ldots, B_k\}$ at $u$. Denote this set by $X$. We then replace each formula of the form $R_j(t_{k_1}, \ldots, t_{k_n})$ in $X$ by $R_j{}'(u, t_{k_1}, \ldots, t_{k_n})$, each formula of the form $\Diamond R_j(t_{k_1}, \ldots, t_{k_n})$ by $R(u, y) \wedge R_j{}'(y, t_{k_1}, \ldots, t_{k_n})$, where $y$ is a new variable, and each formula of the form $\Box R_j(t_{k_1}, \ldots .t_{k_n})$ by $R_j{}'(v_1, t_{k_1}, \ldots, t_{k_n}) \wedge$

[2]This can be checked in polynomial time.

${R_j}'(v_2, t_{k_1}, \ldots, t_{k_n}) \wedge \ldots \wedge {R_j}'(v_m, t_{k_1}, \ldots, t_{k_n})$, where $v_1, v_2, \ldots, v_m$ are all worlds with the property that $R(u, v_i), 1 \leq i \leq m$; (if we are working with the logic $KD4$ or $S4$, then the symbol $R$ here is understood as its transitive closure $R^*$). Thus, the resulting set contains only classical formulae; denote it by $CFO\{B_1, \ldots, B_k\}$. Denote the set of new variables by $NewVar\{B_1, \ldots, B_k\}$; at the moment we also use $\overline{y}$ to denote this set. It is clear that a tuple $\overline{c}$ of constants in the place of $\overline{x}$ satisfies the set $\{B_1, \ldots, B_k\}$ at $u$ iff it satisfies $\exists \overline{y}\ CFO\{B_1, \ldots, B_k\}$ in the classical relational database $RDB$. To compute such tuples $\overline{c}$ we can use known methods of relational databases. For each found tuple $\overline{c}$, we add the atom $A$ with $\overline{x}$ substituted by $\overline{c}$ to $H(u)$.

In the following we present, for example, an algorithm that for a given MDatalog program $P$ in a modal logic $L \in \{KD, T, KDB, B, KD5, KD45, S5\}$ will construct the least $L$-model of $P$. The algorithm is an extended version of Algorithm 3.1. The constructed model will be a propositional $L$-model over the language that takes Herbrand base as the set of primitive propositions; however, we can treat it as a first-order $L$-model. Data structures $W$, $R$, $H$, and $h$ have the same meanings as for Algorithm 3.1. The procedure $CreateEmptyTail_L(x)$ remains unchanged.

For $M = \langle W, \tau, R, H \rangle$ being a model graph, let $RDB(M)$ be the set of classical atoms defined as follows:

$$\begin{aligned} RDB(M) = \{&{R_j}'(u, c_{k_1}, \ldots, c_{k_n}) \mid R_j(c_{k_1}, \ldots, c_{k_n}) \in H(u)\} \\ &\bigcup \{R(x, y) \mid x \in W \wedge y \in W \wedge R(x, y)\}. \end{aligned}$$

The following algorithm differs from Algorithm 3.1 only by the step 2a and the first command of the step 1.

**Algorithm 8.1.**
*Input:* An MDatalog program $P$
in a modal logic $L \in \{KD, T, KDB, B, KD5, KD45, S5\}$
*Output:* The least $L$-model $M = \langle W, \tau, R, h \rangle$ of $P$.
*Steps:*

1. Change every rule $\forall \overline{x}\ \Box^s(A \leftarrow B_1, \ldots, B_k)$ in $P$
   to the form $\Box^s\ \forall \overline{x}\ (A \leftarrow B_1, \ldots, B_k)$.
   Set $W = \{\tau\}$, $H(\tau) = P$, $R = \emptyset$.
   $CreateEmptyTail_L(\tau)$.
   Set $R = Ext_L(R)$.

2. For every world $u \in W$ with $H(u)$ not empty, and for every formula $\phi \in H(u)$:

   (a) Case $\phi = \forall \overline{x}\ (A \leftarrow B_1, \ldots, B_k)$, where $k \geq 1$ :

   - If the vector $\overline{x}$ is empty and $M, u \vDash B_i$ for all $1 \leq i \leq k$, then set $H(u) = H(u) \cup \{A\}$.
   - If the vector $\overline{x}$ is not empty then for every tuple $\overline{c}$ of constant symbols that in the place of $\overline{x}$ satisfies $\exists \overline{y}\ CFO\{B_1, \ldots, B_k\}$ in $RDB(M)$, where $\overline{y} = NewVar\{B_1, \ldots, B_k\}$, add $A$ with $\overline{x}$ substituted by $\overline{c}$ to $H(u)$.

   (b) Case $\phi = \Box\psi$: For every world $v \in W$ satisfying $R(u, v)$, set $H(v) = H(v) \cup \{\psi\}$.

   (c) Case $\phi = \Diamond\psi$, and $\neg(\exists x\ R(u, x) \wedge \psi \in H(x))$:

Let $v$ be a new world.
Set $W = W \cup \{v\}$, $H(v) = \{\psi\}$, $R = R \cup \{(u, v)\}$.
$CreateEmptyTail_L(v)$.
Set $R = Ext_L(R)$.

3. While some change occurred, repeat the step 2.

### 8.2. An Optimization Technique

In this subsection we present a technique that optimizes both execution time and memory for the algorithms of constructing the least models for MDatalog programs.

Let $L$ be one of the serial modal logics considered in this work. Let $P$ be an MDatalog program and $M = \langle W, \tau, R, H \rangle$ be the model graph constructed by the corresponding algorithm that constructs the least $L$-model for $P$.

For any $w \in W$, let $least\_succ(w)$ be the world specified as follows. If $L \in \{KD, T, KDB, B\}$ then $least\_succ(w)$ is the first world in the chain of empty worlds created for $w$. If $L \in \{KD5, KD45, S5\}$ then $least\_succ(w)$ is $\rho$ in the case $R(\tau, w)$, and $\varpi$ in the case $\neg R(\tau, w) \wedge R^2(\tau, w)$. If $L$ is $KD4$ or $S4$ then $least\_succ(w)$ is the world created by the call $CreateNextMinNode(w)$ - if $StatusW(w) \neq minimal$, and $least\_succ(w) = w$ - otherwise.

There exists a binary relation $r$ such that $M \leq M$ wrt. $r$ and that for any $w \in W$, and $u$ reachable from $w$, $r(least\_succ(w), u)$ holds. Such a relation $r$ can be constructed as in the proofs of the main lemmas of [9]. Intuitively, $least\_succ(w)$ is the least successor of $w$.

We claim that $\forall w, u, v\ least\_succ^n(w) = u \wedge R^n(w, v) \to H(u) \subseteq H(v)$, for $n \geq 1$. This assertion can be proved by induction on $n$ and on the creation timestamp of a selected formula from $H(u)$.

When creating a new world $y$ from $x$, we keep $parent(y) = x$. Now observe that for all $w \in W$, if $n$ is the number such that $parent^n(w) = \tau$, then

$$\bigcup_{i=1}^{n} H(least\_succ^i(parent^i(w))) \subseteq H(w)$$

This leads to the following strategy:

- For any world $w \in W$, *solve* the world $least\_succ(w)$ and its successors before solving other successors of $w$.
- For any world $w \in W$, $w \neq \tau$, treat $H(w)$ as the sum of itself plus the set $\bigcup_{i=1}^{n} H(least\_succ^i(parent^i(w)))$, where $n$ is the number such that $parent^n(w) = \tau$.

This strategy saves us a lot of time and memory from recomputing and storing known facts.

## 9. Examples of Application

### An Example Concerning Belief

We can use modal logics to reason about belief. Between the axioms D, T, 4, 5 and B, the axioms D and 4 are widely accepted for belief. The axiom D : $\Box\phi \to \Diamond\phi$ says that if we believe in $\phi$ then we do not believe that $\phi$ does not hold (note that $\Diamond\phi \equiv \neg\Box\neg\phi$); this axiom can be also interpreted as if we believe in $\phi$ then it is possible that $\phi$ holds. We will then sometimes read the formula $\Diamond\phi$ as "it is possible that $\phi$ holds". The axiom $4 : \Box\phi \rightarrow \Box\Box\phi$ says that if we believe in $\phi$ then we believe that we believe in $\phi$. Thus, the logic *KD*4 is one of desirable logics for reasoning about belief. However, as we know, computational complexity of provability in *KD*4 is very high (PSPACE-complete), and modal logic programming in *KD*4 is also very complex (there are not known explicit and acceptable conditions which guarantee an MDatalog program having the least *KD*4-model with polynomial size).

The axiom $5 : \Diamond\phi \rightarrow \Box\Diamond\phi$ can be written as $\neg\Box\psi \rightarrow \Box\neg\Box\psi$ and interpreted as if we do not believe in $\psi$ then we believe that we do not believe in $\psi$. This axiom is somehow as intuitive as the axiom 4. However, one can say that this axiom is a bit strong, claiming that: if we believe in something then this information is somehow kept in our mind, on the other side, there are a lot of things we do not believe in, and generally there is no information kept in our mind about that we believe that we do not believe in that things. From the viewpoint of modal logic programming, the axiom 5 is very useful due to the fact that for *L* being one of the logics *KD*5, *KD*45, and *S*5, any positive propositional modal logic program has the least *L*-model with size bounded by a polynomial. For this reason, the modal logic *KD*45 is an attractive logic for reasoning about belief.

**Example 9.1.** Consider the MDatalog program given below, in which quantifiers are omitted. The first rule says that if $X$ has graduated then we believe that he is employed; the second rule says that we believe that if $X$ lives in a developed country and is employed then we believe that he has a car; the third rule says that we believe that if $X$ is employed then it is possible that $X$ has a car. The next five rules say that we believe in the facts we have. Given some facts, the queries are: "Who is believed to have a car ?" and "Who possibly has a car ?".

$P = \{$
$\Box employed(X) \leftarrow graduated(X),$
$\Box[\Box has\_a\_car(X) \leftarrow employed(X), lives\_in\_a\_developed\_country(X)],$
$\Box[\Diamond has\_a\_car(X) \leftarrow employed(X)],$
$\Box graduated(X) \leftarrow graduated(X),$
$\Box student(X) \leftarrow student(X),$
$\Box employed(X) \leftarrow employed(X),$
$\Box lives\_in\_a\_developed\_country(X) \leftarrow lives\_in\_a\_developed\_country(X),$
$\Box has\_a\_car(X) \leftarrow has\_a\_car(X),$
$graduated(John), graduated(Tomek), student(Kamil),$
$lives\_in\_a\_developed\_country(John), has\_a\_car(Jacek)$
$\}$
$?X \;\; \Box has\_a\_car(X)$
$?X \;\; \Diamond has\_a\_car(X)$

We first consider the program $P$ in the modal logic *KD*45. The *KD*45-model graph that Algorithm 8.1 builds for $P$ is specified as follows:

$W = \{\tau, \rho, \varpi, u\}$, $R = W \times \{\rho, \varpi, u\}$,
$H(\tau) = P \cup \{$
$\Box employed(John), \Box employed(Tomek),$

$\Box graduated(John), \Box graduated(Tomek), \Box student(Kamil),$
$\Box lives\_in\_a\_developed\_country(John), \Box has\_a\_car(Jacek)$
$\}$
$H(\varpi) = H(\rho) = \{$
$[\Box has\_a\_car(X) \leftarrow employed(X), lives\_in\_a\_developed\_country(X)],$
$\Diamond has\_a\_car(X) \leftarrow employed(X),$
$employed(John), employed(Tomek),$
$graduated(John), graduated(Tomek), student(Kamil),$
$lives\_in\_a\_developed\_country(John), has\_a\_car(Jacek),$
$\Box has\_a\_car(John), has\_a\_car(John),$
$\Diamond has\_a\_car(John), \Diamond has\_a\_car(Tomek)$
$\}$
$H(u) = H(\rho) \cup \{has\_a\_car(Tomek)\}$

This model graph forms the least *KD*45-model of $P$, which has the same frame as the model graph and is specified by:

$h(\tau) = \{$
$graduated(John), graduated(Tomek), student(Kamil),$
$lives\_in\_a\_developed\_country(John), has\_a\_car(Jacek)$
$\}$
$h(\varpi) = h(\rho) = h(\tau) \cup \{employed(John), employed(Tomek), has\_a\_car(John)\}$
$h(u) = h(\rho) \cup \{has\_a\_car(Tomek)\}$

To find answers for the first query $?X\ \Box has\_a\_car(X)$, we generate ground instances of this formula and check whether they are satisfied in the constructed model. Answers for this query are ‘Jacek’ and ‘John’. In the same way, answers for the second query $?X\ \Diamond has\_a\_car(X)$ are ‘Jacek’, ‘John’, and ‘Tomek’. Note that these answers follow from the program $P$ *in the logic KD45*.

Now consider the program $P$ in the modal logic *KD*4. The *KD*4-model graph that the first-order version of Algorithm 3.5 builds for $P$ is specified as follows:

$W = \{\tau, 1, 2, 3, 4, 5, 6\},$
$Permanent = \{(\tau, 1), (1, 2), (1, 3), (3, 4), (1, 5), (5, 6)\},$
$Minimal = \{w \in W \mid StatusW(w) = minimal\} = \{2, 4, 6\},$
$R = Permanent \cup \{(2, 2), (4, 4), (6, 6)\} \cup \{2, 3, 4, 5, 6\} \times \{3, 5\},$
$Next(x, \Diamond has\_a\_car(John)) = 3$, for $x \in \{1, 2, 3, 4, 5, 6\}$,
$Next(x, \Diamond has\_a\_car(Tomek)) = 5$, for $x \in \{1, 2, 3, 4, 5, 6\}$,
$H(\tau) = P \cup \{$
$\Box employed(John), \Box employed(Tomek),$
$\Box graduated(John), \Box graduated(Tomek), \Box student(Kamil),$
$\Box lives\_in\_a\_developed\_country(John), \Box has\_a\_car(Jacek)$
$\}$
$H(1) = \{$
$\Box[\Box has\_a\_car(X) \leftarrow employed(X), lives\_in\_a\_developed\_country(X)],$
$\Box[\Diamond has\_a\_car(X) \leftarrow employed(X)],$
$[\Box has\_a\_car(X) \leftarrow employed(X), lives\_in\_a\_developed\_country(X)],$

$\Diamond has\_a\_car(X) \leftarrow employed(X)$,
$\Box employed(John), \Box employed(Tomek)$,
$\Box graduated(John), \Box graduated(Tomek), \Box student(Kamil)$,
$\Box lives\_in\_a\_developed\_country(John), \Box has\_a\_car(Jacek)$,
$employed(John), employed(Tomek)$,
$graduated(John), graduated(Tomek), student(Kamil)$,
$lives\_in\_a\_developed\_country(John), has\_a\_car(Jacek)$,
$\Box has\_a\_car(John)$,
$\Diamond has\_a\_car(John), \Diamond has\_a\_car(Tomek)$
$\}$
$H(6) = H(4) = H(3) = H(2) = H(1) \cup \{has\_a\_car(John)\}$
$H(5) = H(2) \cup \{has\_a\_car(Tomek)\}$

This model graph forms the least *KD*4-model of $P$, which has the same frame as the model graph and is specified by:

$h(\tau) = \{$
$graduated(John), graduated(Tomek), student(Kamil)$,
$lives\_in\_a\_developed\_country(John), has\_a\_car(Jacek)$
$\}$
$h(1) = h(\tau) \cup \{employed(John), employed(Tomek)\}$
$h(6) = h(4) = h(3) = h(2) = h(1) \cup \{has\_a\_car(John)\}$
$h(5) = h(2) \cup \{has\_a\_car(Tomek)\}$

Using this model, the unique answer for the query $?X \;\; \Box has\_a\_car(X)$ is 'Jacek'; answers for the query $?X \;\; \Diamond has\_a\_car(X)$ still are 'Jacek', 'John', and 'Tomek'; answers for the query $?X \;\; \Box\Box has\_a\_car(X)$ are 'Jacek' and 'John'. These answers follow from the program $P$ *in the logic KD4*.

### An Example Concerning Knowledge

One crucial thing that makes knowledge different to belief is that knowledge is true while belief may be false. The axiom T : $\Box\phi \rightarrow \phi$ is thus accepted for knowledge. The axiom 4 : $\Box\phi \rightarrow \Box\Box\phi$ is also admissible.

In the case of belief, we can reason about what we believe in, but we will rather not reason about *our* knowledge, because what is true (known) to us is our knowledge. It makes sense, however, to reason about mutual knowledge; for example, we can reason about what the person $A$ knows, and if he knows what we know, etc. To reason about mutual knowledge we need to use a multi-modal logic. With a single modality, in this subsection, we will give only a simple example.

Suppose that John is a friend of Peter, whose parents are Christ and Jane. Suppose that John knows Peter's father, but there is no information about whether he knows Peter's mother or not. On the question "Who does John know as Peter's parent", one would expect only 'Christ'. How can we formulate this situation ?

In the following, let $\Box$ stand for "John knows". We formulate the above situation by:

$\Box[parent(X, Y) \leftarrow father(X, Y)]$,
$\Box[parent(X, Y) \leftarrow mother(X, Y)]$,

$father(Christ, Peter), mother(Jane, Peter),$
$\Box father(Christ, Peter)$

$?X \;\; \Box parent(X, Peter)$

The *S*4-model graph that the first-order version of Algorithm 3.5 builds for this program is specified by:

$W = \{\tau, \upsilon\}$, $R = Permanent = \{(\tau,\tau), (\tau,\upsilon), (\upsilon,\upsilon)\}$,
$Minimal = \{w \in W \mid StatusW(w) = minimal\} = \{\upsilon\}$,

$H(\tau) = P \cup \{$
$parent(X, Y) \leftarrow father(X, Y),$
$parent(X, Y) \leftarrow mother(X, Y),$
$parent(Christ, Peter), parent(Jane, Peter)$
$\}$
$H(\upsilon) = \{$
$\Box[parent(X, Y) \leftarrow father(X, Y)],$
$\Box[parent(X, Y) \leftarrow mother(X, Y)],$
$parent(X, Y) \leftarrow father(X, Y),$
$parent(X, Y) \leftarrow mother(X, Y),$
$\Box father(Christ, Peter),$
$father(Christ, Peter), parent(Christ, Peter)$
$\}$

The least *S*4-model for the program then has the same frame as the model graph and is specified by:

$h(\tau) = \{$
$father(Christ, Peter), mother(Jane, Peter),$
$parent(Christ, Peter), parent(Jane, Peter)$
$\}$
$h(\upsilon) = \{father(Christ, Peter), parent(Christ, Peter)\}$

Thus, there is only one answer for the query $?X \;\; \Box parent(X, Peter)$ (in *S*4), and it is 'Christ'.

## 10. Conclusions

We have defined the modal query language MDatalog and established conditions that give PTIME complexity for computing queries. Algorithms of constructing the least models for MDatalog programs have been given. They form a bottom-up method for computing queries in MDatalog. We have also demonstrated how MDatalog can be used to reason about belief and knowledge.

## Acknowledgements

This work is a part of my Ph.D. thesis [10]. I would like to thank my supervisor, prof. Andrzej Szałas, for his guidance, and the reviewers, prof. Witold Łukaszewicz and Wojciech Penczek, for comments. I would like also to thank the anonymous reviewer for helpful comments.

## References

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

[2] Apt, K.: Logic Programming, in: *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics* (J. van Leeuwen, Ed.), Elsevier, 1990.

[3] Fariñas del Cerro, L., Penttonen, M.: A Note on the Complexity of the Satisfiability of Modal Horn Clauses, *Logic Programming*, **4**, 1987, 1–10.

[4] Chellas, B.: *Modal Logic: An Introduction*, Cambridge Univ. Press, 1980.

[5] Chen, C., Lin, I.: The Computational Complexity of the Satisfiability of Modal Horn Clauses for Modal Propositional Logics, *Theoretical Computer Science*, **129**, 1994, 95–121.

[6] Hughes, G., Cresswell, M.: *An Introduction to Modal Logic*, Methuen, 1968.

[7] Kripke, S.: Semantical Analysis of Modal Logic, I. Normal Modal Propositional Calculus, *Z. Math. Logic Grundlag*, **9**, 1963, 67–96.

[8] Minker, J.: Logic and Databases: A 20-Year Retrospective, *International Workshop LID'96*, Verlag Springer, 1996.

[9] Nguyen, L.: Constructing the Least Models for Positive Modal Logic Programs, *Fundamenta Informaticae*, **42**(1), 2000, 29–60.

[10] Nguyen, L.: *Results on Modal Reasoning with Applications to Modal Deductive Databases*, Ph.D. thesis, University of Warsaw, 2000.