The Data Complexity of MDatalog in Basic Modal Logics

Linh Anh Nguyen

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

Abstract. We study the data complexity of the modal query language MDatalog and its extension eMDatalog in basic modal logics. MDatalog is a modal extension of Datalog, while eMDatalog is the general modal Horn fragment with the allowedness condition. As the main results, we prove that the data complexity of MDatalog and eMDatalog in $K4$, $KD4$, and $S4$ is PSPACE-complete, in K is coNP-complete, and in KD, T, KB, KDB, and B is PTIME-complete.

1 Introduction

Modal logics can be used to reason about knowledge and belief. It is desirable to study modal extensions of deductive databases. First tries in this direction were done in our previous works [9,10] (modal Datalog defined in [5, Definition 23] is completely different, as it is formulated in classical logic and uses only unary or binary predicates). In [9], we extended Datalog for monomodal logics, giving two languages: MDatalog and eMDatalog. The first one is a natural extension of Datalog, while the second one is the general modal Horn fragment with a refined condition of allowedness. It was shown in [9] that MDatalog and eMDatalog have the same expressiveness in normal monomodal logics. In [10], we studied an extension of MDatalog for multimodal logics of belief and presented bottom-up computational methods for (multi)modal deductive databases.

It is well known that the data complexity of Datalog is complete in PTIME (see, e.g., [6]). In [10], we proved that the data complexity of MDatalog in some multimodal logics of belief which are extensions of $KD45$ is in PTIME. In [9], we gave sufficient conditions for MDatalog in 13 basic monomodal logics (except K and $K4$) to obtain PTIME complexity for computing queries. The complexity results of [9, Theorem 7.3] are not formulated in the way of the data complexity and the relation between them is not close.

In this paper, we study the data complexity of MDatalog and eMDatalog in all of the 15 basic monomodal logics, which are extensions of the logic K using any combination of axioms D, T, B, 4, and 5. The data complexity of MDatalog and eMDatalog is related to the complexity of the propositional modal Horn fragment. In [3], Fariñas del Cerro and Penttonen showed that the satisfiability problem of sets of modal Horn clauses in $S5$ is decidable in PTIME. In [1], Chen and Lin showed that the similar problem for a normal monomodal logic L being an extension of $K5$ (write $K5 \leq L$) is also decidable in PTIME. Chen and Lin also proved that for a normal modal logic L such that $K \leq L \leq S4$ or $K \leq L \leq B$, in particular, for $L \in \{K, KD, KB, KDB, B, K4, KD4, S4\}$, the problem is PSPACE-hard. They also made a comment that the problem is still PSPACE-hard for $S4$ even when the modal depth is restricted to 2. In [8], we showed that the complexity of the satisfiability problem of sets of modal Horn clauses with finitely bounded modal depth in KD, T, KB, KDB, and B is decidable in PTIME. These PTIME results can further be categorized as PTIME-complete. In [11], we showed that the satisfiability problem of sets of modal Horn clauses with modal depth bounded by $k \geq 2$ in the modal logics $K4$ and $KD4$ is PSPACE-complete, and in K is NP-complete.

In this paper we prove that, for L being any one of the 15 basic monomodal logics, the data complexity of MDatalog and eMDatalog in L is the same as the complexity of the unsatisfiability problem in L of the propositional modal Horn fragment with modal depth bounded by $k \geq 2$. This means that the data complexity of MDatalog and eMDatalog in $K4$, $KD4$, and $S4$ is PSPACE-complete, in K is coNP-complete, and in the remaining logics is PTIME-complete.

2 Preliminaries

2.1 Definitions for Modal Logics

The language for propositional modal logics extends the language of classical propositional logic with two modal operators $\Box$ and $\Diamond$.

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. A *propositional Kripke model*, sometimes briefly called a model, is a tuple $\langle W, \tau, R, h \rangle$, where $\langle W, \tau, R \rangle$ is a Kripke frame and h is a function that maps each world of W to a set of primitive propositions. A *model graph* is a tuple $\langle W, \tau, R, H \rangle$, where $\langle W, \tau, R \rangle$ is a Kripke frame and H is a function that maps each world of W to a formula set.

Given a Kripke model $M = \langle W, \tau, R, h \rangle$ and a world $w \in W$, the *satisfaction relation* $\models$ is defined as usual for the classical connectives with two extra clauses for the modal operators as below:

$$\begin{array}{ll} M, w \models \Box\varphi & \text{iff } \forall v \in W.\ R(w, v) \text{ implies } M, v \models \varphi \\ M, w \models \Diamond\varphi & \text{iff } \exists v \in W.\ R(w, v) \text{ and } M, v \models \varphi. \end{array}$$

We say that φ is *satisfied at w in M* if $M, w \models \varphi$. We say that φ is *satisfied in M* and call M a *model of φ* if $M, \tau \models \varphi$.

If we consider all Kripke models, with no restrictions on R, we obtain a normal propositional modal logic with a standard Hilbert-style axiomatization K. Other normal propositional modal logics are obtained by adding to K certain axioms. The most popular axioms used for extending K are D, T, B, 4, and 5, which respectively correspond to seriality ($\forall x\, \exists y.R(x, y)$), reflexiveness, symmetry, transitiveness, and euclideaness ($\forall x\, \forall y\, \forall z.R(x, y) \wedge R(x, z) \rightarrow R(y, z)$) of the accessibility relation. The names of normal propositional modal logics often consist of K and the names of the added axioms, e.g. KDB is the logic which extends K with the axioms D and B. The special cases are T, B, $S4$, and $S5$, which stand for KT, KTB, $KT4$, and $KT5$, respectively.

We refer to the properties of the accessibility relation of a modal logic L as the *L-frame restrictions*. A Kripke model M is an *L-model* if the accessibility relation of M satisfies all L-frame restrictions. We say that φ is *L-satisfiable* if there exists an L-model of φ. We write $\Gamma \models_L \varphi$ to denote that φ is satisfied in every L-model of Γ.

The *modal depth* of a formula φ is the maximal nesting depth of modal operators occurring in φ; e.g., the modal depth of $p \wedge \Box(\Diamond q \vee \Diamond r)$, where p, q, r are primitive propositions, is 2.

The *length* of a formula φ is the total number of symbols occurring in φ. The *size* of a formula set is the sum of the lengths of its formulas.

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

In this work, we consider also first-order modal logics. We restrict to first-order modal logics with *fixed-domain* (i.e. all possible worlds in a first-order Kripke model have the same domain) and *rigid terms* (i.e. the semantics of terms does not depend on possible worlds). As usual for database models, we assume that the signature contains predicate symbols and constant symbols, but no function symbols. The definitions given for propositional modal logics can be shifted in a natural way for first-order modal logics.

We refer to [2,4] for further reading on modal logics.

2.2 The Modal Query Languages MDatalog and eMDatalog

An *MDatalog program clause* is a formula of the form

$$\forall x_1 \ldots \forall x_k \, \Box^h (B_1 \wedge \ldots \wedge B_l \to A)$$

where $\Box^h$ is a sequence of h operators $\Box$, $h \geq 0$, $l \geq 0$, and

- A, B_1, ..., B_l are of the form $\Box E$, $\Diamond E$, or E, with E being a classical atom (recall that a classical atom is a formula of the form $p(t_1, \ldots, t_n)$),
- $x_1, \ldots, x_k$ are all variables occurring in $(B_1 \wedge \ldots \wedge B_l \to A)$,
- all variables of A occur also in $B_1 \wedge \ldots \wedge B_l$.

The last condition in the above definition is usually called *allowedness* (or *range-restrictedness*). It implies that if $l = 0$ then $k = 0$ and A is a ground formula. We will write the above program clause in the form $\Box^h(A \leftarrow B_1, \ldots, B_l)$. We will also write $\varphi \leftarrow \psi$ for $\psi \to \varphi$.

An *MDatalog program* is a set of MDatalog program clauses.

We now define an extension of MDatalog called *eMDatalog*, which allows more sophisticated program clauses.

A formula is *positive* if it does not contain $\neg$ and $\leftarrow$. A formula φ is called a *non-negative modal Horn formula* iff one of the following conditions holds:

- φ is a classical atom;
- φ is of the form $\psi \leftarrow \zeta$, where ψ is a non-negative modal Horn formula and ζ a positive formula without quantifiers such that if $\zeta_1 \vee \zeta_2$ is a subformula of ζ then ζ_1 and ζ_2 have the same variables[1];
- $\varphi = \Box\psi$, or $\varphi = \Diamond\psi$, or $\varphi = \psi \wedge \zeta$, where ψ and ζ are non-negative modal Horn formulas.

In the following, E denotes a classical atom and $Var(\varphi)$ denotes the set of variables of φ. Define the constraint $allowed(\varphi, V)$ for a non-negative modal Horn formula φ and a set of variables V recursively as follows:

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

An *eMDatalog program clause* is a formula of the form $\forall x_1 \ldots \forall x_k\, \varphi$, where φ is a non-negative modal Horn formula, $x_1, \ldots, x_k$ are all variables of φ, and the constraint $allowed(\varphi, \emptyset)$ is true. Observe that an eMDatalog program clause not containing $\leftarrow$ is a ground formula.

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

For $\overline{x}$ being a tuple of variables $(x_1, \ldots, x_k)$, we write $\exists\overline{x}\ \varphi(\overline{x})$ to denote the formula $\exists x_1 \ldots \exists x_k\, \varphi$ and assume that $x_1, \ldots, x_k$ are variables occurring in φ. In that case, for $\overline{c}$ being a tuple of constant symbols $(c_1, \ldots, c_k)$, we write $\varphi(\overline{c})$ to denote the formula obtained from φ by substituting each x_i by c_i, for $1 \leq i \leq k$.

A *query* to an MDatalog/eMDatalog program is a formula of the form $\exists\overline{x}\ \varphi(\overline{x})$, where $\overline{x}$ is a tuple of all variables of φ (which can be empty) and φ is a positive formula without quantifiers such that if $\zeta_1 \vee \zeta_2$ is a subformula of φ then ζ_1 and ζ_2 have the same variables[2].

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

The following proposition states that MDatalog has the same expressiveness as eMDatalog. It was proved in [9] for the case without $\vee$ in non-negative modal Horn formulas. The proof for the extension with $\vee$ is straightforward.

Proposition 1. *For any eMDatalog program P, there exists an MDatalog program P' such that for any query $\exists\overline{x}\ \varphi(\overline{x})$ in the language of P, the answers w.r.t. P' are exactly the answers w.r.t. P. Moreover, P' can be computed from P in polynomial time and the modal depth of P' is equal to the modal depth of P.*

For example, the eMDatalog program $\{\Diamond(p(a) \wedge q(a))\}$ can be transformed to the MDatalog program $\{\Diamond r,\ \Box(p(a) \leftarrow r),\ \Box(q(a) \leftarrow r)\}$.

[1] This extends the corresponding definition given in [9] with $\vee$.
[2] This condition was not considered in [9].

2.3 Modal Deductive Databases and Data Complexity

An MDatalog/eMDatalog program clause is either a rule or a fact. A *rule* is a program clause containing $\leftarrow$, while a *fact* does not contain $\leftarrow$. Note that an MDatalog fact is a ground formula of the form $\Box^h A$, where $h \geq 0$ and A is a formula of the form $\Box E$, $\Diamond E$, or E with E being a classical atom. Observe also that an eMDatalog fact is a positive ground formula not containing $\vee$.

A predicate p is defined by an MDatalog/eMDatalog program clause φ if p appears in φ in the left hand side of all the occurrences of $\leftarrow$.

A modal deductive database can be specified by an MDatalog/eMDatalog program. It can be divided into two parts: an *extensional part* and an *intensional part*. The extensional part consists of facts defining so called *extensional predicates*. The intensional part consists of rules and facts defining so called *intensional predicates*, which are not extensional predicates.

When measuring the "data complexity" of a query language for deductive databases, the query to the program specifying the database is grouped with the intensional part of the database and treated as a fixed *query*, while the extensional part of the database is treated as input. Suppose that we are given an MDatalog (resp. eMDatalog) program P representing a modal deductive database and a query $\exists \overline{x}\, \varphi(\overline{x})$. Denote the extensional part of the database by D and the intensional part by P'. We call the pair $(P', \exists \overline{x}\, \varphi(\overline{x}))$ an *MDatalog* (resp. *eMDatalog*) *query* and D an *extensional MDatalog* (resp. *eMDatalog*) *database instance* (*edb instance* for short).

We say that the *data complexity* of MDatalog (resp. eMDatalog) in a modal logic L is in a complexity class $\mathcal{C}$ if the recognition problem of checking whether $P' \cup D \models_L \varphi(\overline{c})$ with D taken as input is in the complexity class $\mathcal{C}$ for every MDatalog (resp. eMDatalog) query $(P', \exists \overline{x}\, \varphi(\overline{x}))$ and every tuple $\overline{c}$ of constant symbols.

We say that the data complexity of MDatalog (resp. eMDatalog) is complete in a complexity class $\mathcal{C}$, or $\mathcal{C}$*-complete*, if it is in $\mathcal{C}$ and there *exist* an MDatalog (resp. eMDatalog) query $(P', \exists \overline{x}\, \varphi(\overline{x}))$ and a tuple $\overline{c}$ of constant symbols such that the problem of checking whether $P' \cup D \models_L \varphi(\overline{c})$ with D taken as input is $\mathcal{C}$-complete.

It is not clear whether the data complexity of eMDatalog is the same as the data complexity of MDatalog in every modal logic. The problem is that when transforming an eMDatalog program to an MDatalog program, we introduce new rules, which causes that the resulting "query" depends on the input. For this reason, we will study the data complexity of both MDatalog and eMDatalog.

3 Simple Cases

In this section, we show that the data complexity of MDatalog and eMDatalog in $K5$, $KD5$, $K45$, $KD45$, $KB5$, and $S5$ is PTIME-complete, and in $K4$, $KD4$, and $S4$ is in PSPACE.

Let $(P, \exists \overline{x}\, \varphi(\overline{x}))$ be a fixed MDatalog/eMDatalog query, $\overline{c}$ be a fixed tuple of constant symbols for substituting $\overline{x}$, and D be an input *edb* instance with size n.

Let P' be the set of all ground instances of program clauses of P using the constant symbols occurring in P, $\overline{c}$, and D. It is easily seen that the size of P' is bounded by a polynomial of n. Observe that $P \cup D \models_L \varphi(\overline{c})$ iff $P' \cup D \models_L \varphi(\overline{c})$ iff $P' \cup D \cup \{\neg\varphi(\overline{c})\}$ is L-unsatisfiable. The latter set can be treated as a set of propositional formulas. It consists of so called *Horn formulas.* By the results of [3,1], checking whether $P' \cup D \cup \{\neg\varphi(\overline{c})\}$ is L-unsatisfiable is decidable in PTIME (w.r.t. n) for $L \in \{K5, KD5, K45, KD45, KB5, S5\}$.[3] By Ladner [7], for $L \in \{K4, KD4, S4\}$, checking whether $P' \cup D \cup \{\neg\varphi(\overline{c})\}$ is L-unsatisfiable is decidable in PSPACE (the Horn property is not important here). We arrive at:

Theorem 1. *The data complexity of MDatalog and eMDatalog in K5, KD5, K45, KD45, KB5, and S5 is PTIME-complete.*

The lower bound follows from that the data complexity of Datalog is complete in PTIME (see, e.g., [6]).

Lemma 1. *The data complexity of MDatalog and eMDatalog in K4, KD4, and S4 is in PSPACE.*

4 The Data Complexity of MDatalog in *K*, *K*4, *KD*4, *S*4

In this section, we show that the data complexity of MDatalog and eMDatalog in $K4$, $KD4$, and $S4$ is PSPACE-complete, and in K is coNP-complete.

Lemma 2. *Every finite set X of propositional modal Horn clauses can be transformed in PTIME to a set Y of propositional modal Horn clauses such that:*

- *Y contains at most one negative clause, which is of the form $\leftarrow p$;*
- *each non-negative clause of Y contains no more than 3 literals;*
- *the modal depth of Y is the same as the modal depth of X;*
- *Y is L-satisfiable iff X is L-satisfiable, for any normal modal logic L.*

Proof. Let p be a fresh primitive proposition. Replace each negative clause $\varphi_i = \Box^s(\leftarrow B_1, \ldots, B_k)$ of X by the following clauses:

$$\Box^s(p_{i,s} \leftarrow B_1, \ldots, B_k), \quad \Box^{s-1}(p_{i,s-1} \leftarrow \Diamond p_{i,s}), \quad \ldots, \quad p \leftarrow \Diamond p_{i,1}$$

where $p_{i,s}, \ldots, p_{i,1}$ are fresh primitive propositions. (If $s = 0$ then φ_i is replaced by $p \leftarrow B_1, \ldots, B_k$.) Then add to the obtained set the clause $\leftarrow p$. Denote the resulting set by X'.

Next, replace each non-negative clause $\varphi_i = \Box^s(A \leftarrow B_1, \ldots, B_k)$ of X', where $k > 2$, by the following clauses:

$$\begin{array}{l} \Box^s(p_{i,2} \leftarrow B_1, B_2) \\ \Box^s(p_{i,3} \leftarrow p_{i,2}, B_3) \\ \ldots \\ \Box^s(p_{i,k} \leftarrow p_{i,k-1}, B_k) \\ \Box^s(A \leftarrow p_{i,k}) \end{array}$$

[3] The definitions of Horn clauses in [3,1] are more restrictive than our definition of Horn formulas [8]. However, every set of Horn formulas can be transformed in polynomial time to a set of Horn clauses that preserves satisfiability.

where $p_{i,2}, \ldots, p_{i,k}$ are fresh primitive propositions. Let Y be the resulting set. It is easy to verify that Y satisfies the assertions of the lemma.

Corollary 1. *Let X be a set of propositional modal Horn clauses such that: the modal depth of X is not greater than 2, X contains at most one negative clause, which is of the form $\leftarrow p$, and each non-negative clause of X contains no more than 3 literals. Then the problem of checking whether X is L-satisfiable is NP-complete for $L = K$ and PSPACE-complete for $L \in \{K4, KD4, S4\}$.*

Proof. This corollary immediately follows from Lemma 2 and the results of [11,1] that the satisfiability problem of sets of propositional modal Horn clauses with modal depth bounded by $k \geq 2$ in K is NP-complete, in $K4$, $KD4$, and $S4$ is PSPACE-complete.

Lemma 3. *Let X be as in Corollary 1 with $\psi = \leftarrow q$ being the only negative clause. Then there exist an MDatalog query (P, φ), where P does not depend on X and φ is a positive ground formula depending only on ψ, and an* edb *instance D with size in polynomial order in the size of X such that X is L-satisfiable iff $P \cup D \not\models_L \varphi$, where L is any normal modal logic.*

Proof. Each non-negative clause of X is of the form $\Box^s(A \leftarrow B_1, \ldots, B_k)$, where $0 \leq s \leq 2$, $0 \leq k \leq 2$, and $A, B_1, \ldots, B_k$ are of the form p, $\Box p$, or $\Diamond p$. Let K be the number of such possible forms (i.e. $K = 3 \times 3 \times (1 + 3 + 3 \times 3)$).

Suppose that primitive propositions occurring in X are numbered from 1 to n and denoted by $p_1, \ldots, p_n$. We represent each non-negative clause ξ of X by a first-order program clause representing the form of ξ plus a ground first-order atom indicating ξ. For simplicity, we illustrate this using a clause $\xi = \Box\Box(p_i \leftarrow \Box p_j, \Diamond p_k)$. Let $1 \leq t \leq K$ be the number indicating the form of this clause. The clause ξ is then represented by $\Box\Box(p(x) \leftarrow \Box p(y), \Diamond p(z), \mathit{clause}(t, x, y, z))$ and $\Box\Box \mathit{clause}(t, i, j, k)$. Note that the program clause depends only on the form of ξ, while the atom indicates ξ. In this way, the set of non-negative clauses of X is represented by an MDatalog program P and a set D of ground first-order atoms. Furthermore, we can assume that P does not depend on X, as we can include in P an appropriate program clause for *every* $1 \leq t \leq K$.

If $\psi = \leftarrow p_i$ then let $\varphi = p(i)$. It is clear that X is L-satisfiable iff $P \cup D \cup \{\neg\varphi\}$ is L-satisfiable, and iff $P \cup D \not\models_L \varphi$, where L is any normal modal logic.

Theorem 2. *The data complexity of MDatalog and eMDatalog in K4, KD4, and S4 is PSPACE-complete.*

Proof. By Lemma 1, the data complexity of MDatalog and eMDatalog in $K4$, $KD4$, and $S4$ is in PSPACE. Let X be a set of propositional modal Horn clauses as in Lemma 3. By Corollary 1, the problem of checking L-satisfiability of X for $L \in \{K4, KD4, S4\}$ is PSPACE-complete. By Lemma 3, this problem is reducible in polynomial time to a problem of answering a fixed MDatalog query (P, φ) w.r.t. an *edb* instance D depending on X (here, without loss of generality, assume that φ is fixed). Hence, the data complexity of MDatalog and eMDatalog in $K4$, $KD4$, and $S4$ is complete in PSPACE.

Theorem 3. *The data complexity of MDatalog and eMDatalog in K is coNP-complete.*

Proof. We first show that the data complexity of eMDatalog in K is in coNP. Let (P, φ) be a fixed eMDatalog query, where φ is a positive ground formula, and D be an eMDatalog *edb* instance. To answer the query w.r.t. the input D in the logic K is to check whether $P \cup D \models_K \varphi$, or equivalently, whether $P \cup D \cup \{\neg\varphi\}$ is K-unsatisfiable.

Let n be the size of D and c be the modal depth of $P \cup \{\neg\varphi\}$. Let P' be the set of all ground instances of clauses of P using the constant symbols occurring in P, φ, D. The size of P' is bounded by a polynomial of n. Observe that $P \cup D \cup \{\neg\varphi\}$ is K-satisfiable iff $P' \cup D \cup \{\neg\varphi\}$ is K-satisfiable.

Consider the process of constructing a K-model graph for $P' \cup D \cup \{\neg\varphi\}$. At the beginning, the model graph contains only the actual world τ with $P' \cup D \cup \{\neg\varphi\}$ in the negative normal form as the content. Then for every world w and every formula ψ from the content of w, we realize ψ at w as follows. If $\psi = \zeta_1 \vee \ldots \vee \zeta_k$ then nondeterministically choose some ζ_i and add ζ_i to the content of w. If $\psi = \Diamond\zeta$ then create a new empty world u, connect w to u via the accessibility relation, and add ζ to the content of u. If $\psi = \Box\zeta$ then add ζ to the content of every world accessible from w. If at the end we obtain a model graph which is saturated (in the sense that, for every w, every formula ψ in the content of w has been realized at w) and is consistent (in the sense that no world contains both p and $\neg p$ for some primitive proposition p), then $P' \cup D \cup \{\neg\varphi\}$ is K-satisfiable. Observe that if a world w in the constructed model graph is inconsistent (i.e. w contains both p and $\neg p$ for some p), then the path from τ to w via the accessibility relation is not longer than c, since D contains only positive formulas. This means that when checking consistency of the constructed model graph, we need to pay attention only to worlds not far away from τ than c edges. The total size of that fragment of the constructed model graph is bounded by a polynomial of n. Hence, checking whether $P' \cup D \cup \{\neg\varphi\}$ is K-satisfiable can be nondeterministically done in polynomial time. Consequently, the problem of answering the query (P, φ) w.r.t. the input D in the logic K is in the coNP class.

Analogously as in the proof of Theorem 2, using Corollary 1 and Lemma 3 we conclude that the data complexity of MDatalog and eMDatalog in K is complete in coNP.

5 The Data Complexity of eMDatalog in *KD*, *T*, *KB*, *KDB*, and *B*

In this section, we show that the data complexity of MDatalog and eMDatalog in KD, T, KB, KDB, and B is complete in PTIME. For this aim, we first present an algorithm of constructing a "least" L-model for a given eMDatalog program, where $L \in \{KD, T, KDB, B\}$.

We say that a propositional Kripke model M is less than or equal to M' if for every positive propositional formula φ, if $M \models \varphi$ then $M' \models \varphi$. M is called a *least L-model* of an eMDatalog program 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 M is a least L-model of P and φ is a positive propositional formula then $P \models_L \varphi$ iff $M \models_L \varphi$.

In the algorithm given below, as a data structure we have a model graph $M = \langle W, \tau, R, H \rangle$ and a binary relation R' being the skeleton of R. We sometimes refer to M as the propositional model $\langle W, \tau, R, h \rangle$, with $h(x) = \{E \mid E$ is a ground classical atom belonging to $H(x)\}$. We write $M, u \models \varphi$ to denote that φ is true at u in the *model* M.

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

Algorithm 1

```
Input: A ground eMDatalog program $P$ in $L \in \{KD, T, KDB, B\}$
       treated as a set of propositional formulas.
Output: A least $L$-model $M = \langle W, \tau, R, h \rangle$ of $P$.

1. Set $W = \{\tau\}$, $H(\tau) = P$, $R' = \emptyset$.
   $CreateEmptyTail_L(\tau)$.
2. For every $u \in W$ and for every $\varphi \in H(u)$:
   (a) If $\varphi = (\psi \leftarrow \zeta)$ and $M, u \models \zeta$ then set $H(u) = H(u) \cup \{\psi\}$.
   (b) If $\varphi = \psi \wedge \zeta$ then set $H(u) = H(u) \cup \{\psi, \zeta\}$.
   (c) If $\varphi = \Box\psi$ then for every $v \in W$ s.t. $R(u, v)$ set $H(v) = H(v) \cup \{\psi\}$.
   (d) If $\varphi = \Diamond\psi$ and $\neg(\exists x\, R'(u, x) \wedge \psi \in H(x))$ then:
          Let $u_\psi$ be a new world.
          Set $W = W \cup \{u_\psi\}$, $H(u_\psi) = \{\psi\}$, $R' = R' \cup \{(u, u_\psi)\}$.
          $CreateEmptyTail_L(u_\psi)$.
3. While some change occurred, repeat step 2.
```

This algorithm is very similar to Algorithm 5.1 in [8], which constructs a least L-model for a given positive propositional modal logic program. The following lemma can be proved as done for Algorithm 5.1 in [8].

Lemma 4. *The above algorithm always terminates and the constructed model M is a least L-model of P.*

Let $M = \langle W, \tau, R, h \rangle$ be a Kripke model. Define the distance from τ to a world $w \in W$ via R to be the length of a shortest path from τ to w via R (undefined if there does not exist such a path). For $k \geq 0$, we define $M_{|k}$ to be the model obtained from M by restricting it to the worlds with the distance from τ via R not greater then k. The following lemma can be proved easily.

Lemma 5. *Let $M = \langle W, \tau, R, h \rangle$ be a Kripke model and φ a formula with modal depth not greater than k. Then $M \models \varphi$ iff $M_{|k} \models \varphi$.*

If we are given an eMDatalog program P representing a modal deductive database and a positive ground formula φ to check whether $P \models_L \varphi$, then instead of constructing a least L-model M of P to check whether $M \models \varphi$ we can construct only $M_{|k}$ and check whether $M_{|k} \models \varphi$ where k is a number not less than the modal depth of φ. In the following we show that such a restricted model $M_{|k}$ can be constructed without having the whole model M.

Let φ be a positive ground formula and ψ be a subformula of φ with a fixed position. We define the *modal context of ψ in φ* to be the sequence of modal operators occurring in φ such that ψ is under the scope of them. For example, the modal context of $\Box(r \wedge s)$ in $\Diamond(\Box p \wedge \Diamond(\Diamond q \wedge \Box(r \wedge s)))$ is $\Diamond\Diamond$, which consists of the first and the second $\Diamond$. We call a sequence of modal operators a *modality* and denote the empty modality by ε.

For $L \in \{T, KDB, B\}$, let the *set of rules for shrinking modalities in L* consist of: $\Box \to \varepsilon$ if $L \in \{T, B\}$, $\Box\Box \to \varepsilon$ and $\Diamond\Box \to \varepsilon$ if $L \in \{KDB, B\}$. Note that the first rule corresponds to axiom T, while the two latter rules correspond to axiom B. A modality $\triangle$ is *reducible to ε* in $L \in \{T, KDB, B\}$ if ε is derivable from $\triangle$ using the rules for shrinking modalities in L.

Lemma 6. *Consider Algorithm 1 and a moment when a formula ψ is added to $H(x)$. Suppose that the distance from τ to x via R' is greater than the modal depth of every rule of P. Then:*

- *there exists $\Box\psi \in H(y)$ s.t. $R'(y, x)$ holds; or*
- *there exists $\Diamond\psi \in H(y)$ s.t. $x = y_\psi$ (i.e. x is created from y using $\Diamond\psi$); or*
- *before adding ψ to $H(x)$ there exists already $\psi' \in H(x)$ such that ψ is a subformula of ψ' under a modal context which is reducible to ε in L.*

Proof. We prove this lemma by induction on the number of steps needed to add ψ to $H(x)$. The only non-trivial case is when ψ is added to $H(x)$ at Step 2c with $x = v$, $L \in \{KDB, B\}$ and $R'(v, u)$ holds. Consider that case. Applying the induction hypothesis for the moment when $\varphi = \Box\psi$ is added to $H(u)$, we obtain that $\Diamond\varphi \in H(v)$ or $\Box\varphi \in H(v)$ or before adding φ to $H(u)$ there exists already $\varphi' \in H(u)$ such that φ is a subformula of φ' under a modal context which is reducible to ε in L. By repeatedly applying the induction hypothesis for the moment when φ' is added to $H(u)$, we derive that there exists a formula $\xi \in H(u)$ such that φ is a subformula of ξ under a modal context which is reducible to ε and ξ was added to $H(u)$ because $\Diamond\xi \in H(v)$ or $\Box\xi \in H(v)$. It is easily seen that the inductive assertion immediately follows.

Lemma 7. *Consider Algorithm 1. Suppose that $\varphi \in H(u)$ and the distance from τ to u via R' is greater than the modal depth of every rule of P. Then every subformula ψ of φ under a modal context which is reducible to ε will be added to $H(u)$.*

Proof. By induction on the structure of φ.

We now present a modified version of Algorithm 1. The modifications are highlighted by another font.

Algorithm 2

```
Input: A ground eMDatalog program $P$ in $L \in \{KD, T, KDB, B\}$
        treated as a set of propositional formulas.
        A number $k$ greater than the modal depth of every rule of $P$.
Output: A model $M = \langle W, \tau, R, h\rangle$.

1. Set $W = \{\tau\}$, $H(\tau) = P$, $R' = \emptyset$.
   $CreateEmptyTail_L(\tau)$.
2. For every $u \in W$ with $H(u)$ not empty and for every $\varphi \in H(u)$:
   (a) If $\varphi = (\psi \leftarrow \zeta)$ and $M, u \models \zeta$ then set $H(u) = H(u) \cup \{\psi\}$.
   (b) If $\varphi = \psi \wedge \zeta$ then set $H(u) = H(u) \cup \{\psi, \zeta\}$.
   (c) If $\varphi = \Box\psi$ then for every $v \in W$ s.t. $R(u, v)$ set $H(v) = H(v) \cup \{\psi\}$.
   (d) If $\varphi = \Diamond\psi$ and $\neg(\exists x\ R'(u, x) \wedge \psi \in H(x))$ and the path from $\tau$ to $u$ via
       $R'$ is shorter than $k$ then:
          Let $v$ be a new world.
          Set $W = W \cup \{v\}$, $H(v) = \{\psi\}$, $R' = R' \cup \{(u, v)\}$.
          $CreateEmptyTail_L(v)$.
   (e) If $L \in \{T, KDB, B\}$ and the path from $\tau$ to $u$ via $R'$ has length $k$ then, for
       every subformula $\psi$ of $\varphi$ under a modal context which is reducible to $\varepsilon$ in
       $L$, set $H(u) = H(u) \cup \{\psi\}$.
3. While some change occurred, repeat step 2.
```

Lemma 8. *Let P be a ground eMDatalog program s.t. the modal depths of the rules of P are not greater than k. Let M and M' be respectively the outputs of Algorithm 1 and Algorithm 2 for P in $L \in \{KD, T, KDB, B\}$. Then $M_{|k}$ is isomorphic with M'.*

This lemma immediately follows from Lemmas 6 and 7.

Theorem 4. *The data complexity of MDatalog and eMDatalog in $L \in \{KD, T, KDB, B\}$ is PTIME-complete.*

Proof. Since the data complexity of Datalog is complete in PTIME (see, e.g., [6]), it suffices to show that the data complexity of eMDatalog in L is in PTIME.

Let (P', φ) be a fixed eMDatalog query, where φ is a positive ground formula, and D be an eMDatalog *edb* instance. To answer the query w.r.t. the input D in L is to check whether $P' \cup D \models_L \varphi$. Let P'' be the set of all ground instances of clauses of P' using the constant symbols occurring in P', φ, D. Observe that $P' \cup D \models_L \varphi$ iff $P'' \cup D \models_L \varphi$.

Let $P = P'' \cup D$ and let k be the maximum of the modal depths of P' and φ plus 1. Note that k is fixed. Let M be the result of the execution of Algorithm 2 for P and k. By Lemmas 4, 5, and 8, we have that $P'' \cup D \models_L \varphi$ iff $M \models \varphi$.

Let n be the size of D. As the query (P', φ) is fixed, the size of P'' is bounded by a polynomial of n. The size of M and the number of steps needed to construct M are bounded by a polynomial in the size of P''. Checking whether $M \models \varphi$ can be done in polynomial time in the sizes of M and φ. Totally, checking whether $P' \cup D \models_L \varphi$ can be done in polynomial time in the size of the input D, and hence the data complexity of eMDatalog in L is in PTIME.

The remaining logic KB is an *almost serial* (see [9]) modal logic corresponding to B. Using the techniques of [8,9], we can derive the following result.

Corollary 2. *The data complexity of MDatalog and eMDatalog in KB is PTIME-complete.*

6 Conclusions

We have proved that, for L being any one of the 15 basic monomodal logics, the data complexity of MDatalog and eMDatalog in L is the same as the complexity of the unsatisfiability problem in L of the propositional modal Horn fragment with modal depth bounded by $k \geq 2$. This result is a bit surprising, as MDatalog and eMDatalog are fragments of first-order modal logics and can contain data with unbounded modal depth. The eMDatalog language is an interesting fragment of modal logic that has low complexity in KD, T, KB, KDB, and B. On the other hand, the PSPACE-complete data complexity of MDatalog in $K4$, $KD4$, and $S4$ shows that these latter modal logics (and their multimodal extensions) are hard in the view of deductive databases (unless PSPACE = PTIME).

Acknowledgements. I would like to thank the reviewers for useful comments.

References

1. C.C. Chen and I.P. Lin. The computational complexity of the satisfiability of modal Horn clauses for modal propositional logics. *Theor. Comp. Sci.*, 129:95–121, 1994.
2. M.J. Cresswell and G.E. Hughes. *A New Introduction to Modal Logic*. Routledge, 1996.
3. L. Fariñas del Cerro and M. Penttonen. A note on the complexity of the satisfiability of modal Horn clauses. *Logic Programming*, 4:1–10, 1987.
4. M. Fitting and R.L. Mendelsohn. *First-Order Modal Logic*. Springer, 1998.
5. G. Gottlob, E. Grädel, and H. Veith. Linear time datalog and branching time logic. In *Logic-Based Artif. Int.*, pages 443–467. Kluwer Academic Publishers, 2000.
6. Ch. Koch and S. Scherzinger. Lecture notes on database theory. `http://www-db.cs.unisb.de/teaching/dbth0506/slides/dbthdatalog2.pdf`.
7. R. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM Journal of Computing*, 6:467–480, 1977.
8. L.A. Nguyen. Constructing the least models for positive modal logic programs. *Fundamenta Informaticae*, 42(1):29–60, 2000.
9. L.A. Nguyen. The modal query language MDatalog. *Fundamenta Informaticae*, 46(4):315–342, 2001.
10. L.A. Nguyen. On modal deductive databases. In J. Eder, H.-M. Haav, A. Kalja, and J. Penjam, editors, *Proc. of ADBIS 2005, LNCS 3631*, pages 43–57. Springer, 2005.
11. L.A. Nguyen. On the complexity of fragments of modal logics. In R. Schmidt, I. Pratt-Hartmann, M. Reynolds, and H. Wansing, editors, *Advances in Modal Logic - Volume 5*, pages 249–268. King's College Publications, 2005.