On the Deterministic Horn Fragment of Test-free PDL

Linh Anh Nguyen

ABSTRACT. We study the deterministic Horn fragment of test-free propositional dynamic logic ($\mathrm{PDL}^{(0)}$). This fragment adopts the restriction that, in bodies of program clauses and goals, special universal modal operators which are a kind of combination of $\Box$ and $\Diamond$ are used instead of $\Box$. The fragment contains deterministic positive logic programs and deterministic negative clauses, whose negations form serial positive formulae. A least Kripke model for a deterministic positive logic program in $\mathrm{PDL}^{(0)}$ may not exist, because $\mathrm{PDL}^{(0)}$ is a non-serial modal logic. In this work, we present an algorithm that, given a deterministic positive logic program P in $\mathrm{PDL}^{(0)}$, constructs a least pseudo-model of P. A pseudo-model is similar to a Kripke model except that it contains two sets of accessibility relations, one for dealing with existential modal operators and the other for dealing with universal modal operators. A least pseudo-model M of P has the property that, for every serial positive formula φ, $P \models \varphi$ iff $M \models \varphi$. Furthermore, checking whether $M \models \varphi$ is solvable in polynomial time in the sizes of M and φ. Our algorithm runs in exponential time and returns a pseudo-model with size $2^{O(n^2)}$. We give a deterministic positive logic program in $\mathrm{PDL}^{(0)}$ such that every pseudo-model characterizing it must have size $2^{\Omega(n)}$.

KEYWORDS: PDL, finite automata, Horn logic, minimal models

1 Introduction

Modal logic programming extends classical logic programming with modalities. There are two approaches in modal logic programming: the translation approach [3, 10] and the direct approach [1, 2, 8, 9]. In the translation approach, both the functional translation method of Debart et al. [3] and the semi-functional translation method of Nonnengart [10] assume that the base modal logic is serial, i.e. it contains axiom $\Diamond_i \top$ for every modal index i. Using the direct approach for modal logic programming, Balbiani et al. [1] considered only serial modal logics, and in our previous works [7, 8, 9], we considered only serial or almost serial modal logics. In [2] Baldoni et al. studied modal logic programming using the direct approach for grammar logics, which are non-serial normal modal logics with axioms of the form $[t_1] \dots [t_n]\varphi \to [s_1] \dots [s_m]\varphi$. However, they considered only modal logic programs without existential modal operators.

Seriality has thus played an important role in the theory of modal logic programming. It is an essential assumption for the functional and semi-functional translation methods. For the direct approach, let us consider the following program in the modal logic K:

$$\begin{aligned} &\Box p \leftarrow \\ &q \leftarrow \Diamond p \\ &s \leftarrow \Box r \end{aligned}$$

The problem is whether there exists a world accessible from the actual world. If there exists then $\Box p$ implies $\Diamond p$, which then implies q. If there does not then $\Box r$ holds and implies s. The program is thus "nondeterministic" because the accessibility relation is not serial. In the above program, $\Box r$ does not follow from the program, but it may unwantedly become true when there are no worlds accessible from the actual world. To overcome this problem, instead of the program clause $s \leftarrow \Box r$ we can use $s \leftarrow \boxast r$, where $\boxast$ has the semantics defined by $\boxast\varphi \equiv (\Box\varphi \wedge \Diamond\varphi)$ or $\boxast\varphi \equiv (\Box\varphi \wedge \Diamond\top)$. One can say that allowing modal operators like $\boxast$ is not a "solution" for dealing with non-seriality because $\boxast$ contains "seriality" itself. Our justifications for using modal operators like $\boxast$ are as follows:

- If the base modal logic is deliberately chosen to be K, then adopting $\boxast$ is an appropriate solution. Note that we will still allow $\Box$ to appear in contexts and heads of program clauses.

- While seriality is a natural assumption in some applications, e.g. to state that knowledge and belief are consistent, it cannot be assumed in some cases. For example, if a is an action, we may not want a to be always admissible. That is, we may not want to adopt axiom $\langle a\rangle\top$, and in that case $[a]\varphi$ does not imply $\langle a\rangle\varphi$.

- Finally, program clauses like $s \leftarrow \boxast r$ are more acceptable than $s \leftarrow \Box r$. In $s \leftarrow \boxast r$, the premise $\boxast r$ guarantees that $\Box r$ actually follows from the program, while in $s \leftarrow \Box r$, the premise $\Box r$ may "accidentally" be true.

In this work, we study the deterministic Horn fragment of test-free propositional dynamic logic (PDL$^{(0)}$), which is a non-serial modal logic and can express K_n. This fragment adopts the restrictions that: i) in bodies of program clauses and goals, modal operators like $\boxast$ are used instead of universal modal operators like $\Box$; ii) implicit disjunction such as $\langle\pi \cup \pi'\rangle$ and $\langle\pi^*\rangle$ is not allowed in heads of program clauses. The deterministic Horn fragment of PDL$^{(0)}$ contains deterministic positive logic programs and deterministic negative clauses. Negation of a deterministic negative clause is a serial positive formula. In general, a serial positive formula is a positive formula which may contains modal operators like $\boxast$ but not modal operators like $\Box$.

Constructing a least Kripke model for a given positive modal logic program in a serial propositional modal logic L is a useful starting point for developing semantics for positive modal logic programs in the corresponding first-order modal logic L. We have demonstrated this in [7, 8] for the basic serial monomodal logics KD, T, KDB, B, $KD4$, $S4$, $KD5$, $KD45$, $S5$.

In [7], we presented algorithms that, given a positive propositional modal logic program P, construct a least L-model of P, where L is one of the listed modal logics. As a continuation of [7], in [8] we have developed the least model semantics, fixpoint semantics, and SLD-resolution calculi for positive modal logic programs in the corresponding first-order modal logics L.

In PDL$^{(0)}$, a least Kripke model of a deterministic positive logic program P may not exist.[1] However, we can talk about least "pseudo-models" of P. A pseudo-model is similar to a Kripke model except that it contains two sets of accessibility relations, one for dealing with existential modal operators and the other for dealing with universal modal operators. Informally, M is a least pseudo-model of P if it satisfies P and for every pseudo-model M' of P, M is less than or equal to M'. A least pseudo-model M of P has the property that, for every serial positive formula φ, $P \models \varphi$ iff $M \models \varphi$. Furthermore, checking whether $M \models \varphi$ is solvable in polynomial time in the sizes of M and φ.

In this work, we present an algorithm that, given a deterministic positive logic program P, constructs a least pseudo-model of P. Our algorithm runs in exponential time and returns a pseudo-model with size $2^{O(n^2)}$. We give a deterministic positive logic program in PDL$^{(0)}$ such that every pseudo-model characterizing it must have size $2^{\Omega(n)}$.

From the view of the theory of complexity and expressiveness, the deterministic Horn fragment of PDL$^{(0)}$ does not have interesting properties. However, this fragment and our method for it are useful for the following reasons:

1. If a knowledge base is represented by a deterministic positive logic program P and the given query is a serial positive formula φ, then having a least pseudo-model M of P, checking whether $P \models \varphi$ can be reduced to checking whether $M \models \varphi$. This method is especially useful when the knowledge base rarely changes.

2. Our method for answering whether $P \models \varphi$ by constructing a least pseudo-model of P is bottom-up. It does not create choice points, while the traditional tableaux method for this problem would intensively use the "or" splitting rule and we know that a wrong choice when exploring tableaux, e.g. one near the root of the search tree, would cost much. Hence, our bottom-up method is more efficient than the traditional tableaux method, even though both the methods can give an algorithm with EXPTIME complexity.

3. The deterministic Horn fragment of PDL$^{(0)}$ eliminates nondeterminism (of PDL$^{(0)}$). How much important is this property? In [6], Hus-

[1] In [7], we showed that the logic program $\{\Box p\}$ does not have any least K-model. This implies that the deterministic positive logic program $\{[\sigma]p\}$ does not have any least Kripke model in PDL$^{(0)}$. The only reason is the non-seriality of PDL$^{(0)}$. It can be shown that adding the seriality axiom $[\pi]\varphi \to \langle\pi\rangle\varphi$ to PDL$^{(0)}$ causes that every positive logic program without implicit disjunctions $\langle\pi' \cup \pi''\rangle$ and $\langle\pi'^{*}\rangle$ in heads of program clauses has a finite least model in the resulting logic.

tadt et al. proved that the data complexity of query answering in the Horn fragment Horn-$\mathcal{SHIQ}$ of the description logic $\mathcal{SHIQ}$ is in PTIME, while in the full description logic $\mathcal{SHIQ}$ it is complete in coNP.[2] Here, we can also prove that the data complexity of query answering in the deterministic Horn fragment of the PDL$^{(0)}$-like description logic is in PTIME (see Section 6 for more details). This is an interesting property for practical applications. Also note that our deterministic Horn fragment of PDL$^{(0)}$ is more relaxed than the Horn-$\mathcal{SHIQ}$ fragment in the aspect that the constructor $\forall R.C$ (a $[\pi]$-like constructor) is disallowed in bodies of program clauses and queries of Horn-$\mathcal{SHIQ}$, while we allow $[\pi]$ in the form $[\pi]_\diamond$ to appear in bodies of program clauses and queries of the deterministic Horn fragment of PDL$^{(0)}$.

4. As mentioned earlier, we can extend the method of this work for dealing with logic programming in PDL$^{(0)}$.

Our algorithm uses formulae with automaton-modal operators, which are similar to formulae of automaton propositional dynamic logic (APDL) [5]. In [4], Goré and Nguyen also used such formulae for developing analytic tableau calculi with the superformula property for regular grammar logics. In both [4] and this work, formulae with automaton-modal operators are used to record the potentiality inherited from predecessor worlds, which guarantees that when a world w is created from u, the content of w can be computed from the content of u. This technique plays an essential role in constructing finite models.

The rest of this paper is structured as follows. In Section 2, we define PDL$^{(0)}$, the deterministic Horn fragment of PDL$^{(0)}$, automaton-modal operators, pseudo-models, and introduce an ordering of pseudo-models. In Section 3, we present our algorithm. In Section 4, we give characterizations of least pseudo-models of deterministic positive logic programs in PDL$^{(0)}$. In Section 5, we study the lower bound of sizes of such pseudo-models. Further work and concluding remarks are given in Section 6.

2 Preliminaries

2.1 Test-free Propositional Dynamic Logic

The language of test-free propositional dynamic logic (PDL$^{(0)}$) is built from two disjoint sets: Π_0 is a countable set of atomic programs and Φ_0 is a countable set of atomic propositions. We use σ to denote an element of Π_0 and p to denote an element of Φ_0. Programs and formulae are recursively defined using the BNF grammar below:

$$\begin{aligned} \Pi \ni \pi \quad &::= \quad \sigma \mid \pi \cup \pi \mid \pi ; \pi \mid \pi^* \\ \Phi \ni \varphi \quad &::= \quad \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \to \varphi \mid [\pi]\varphi \mid \langle\pi\rangle\varphi \end{aligned}$$

[2] When measuring the data complexity, the TBox of the considered knowledge base is treated as the intensional part, while the ABox is treated as the extensional part.

(The version PDL with test contains also the construction $\varphi?$ as a program.[3]) We use π, α, β to denote elements of Π. A word $\sigma_1 \dots \sigma_k$ over alphabet Π_0 will also be treated as the program $\sigma_1; \dots; \sigma_k$. An operator $[\pi]$ is called a *universal* modal operator, while $\langle\pi\rangle$ is called an *existential* modal operator.

A program of $\mathrm{PDL}^{(0)}$ is a regular expression over the alphabet Π_0. Such an expression π generates a regular language $\mathcal{L}(\pi)$ specified as follows: $\mathcal{L}(\sigma) = \{\sigma\}$, $\mathcal{L}(\pi \cup \pi') = \mathcal{L}(\pi) \cup \mathcal{L}(\pi')$, $\mathcal{L}(\pi;\pi') = \mathcal{L}(\pi).\mathcal{L}(\pi')$, and $\mathcal{L}(\pi^*) = (\mathcal{L}(\pi))^*$, where if L and M are sets of words then $L.M = \{\alpha\beta \mid \alpha \in L, \beta \in M\}$ and $L^* = \bigcup_{n\geq 0} L^n$ with $L^0 = \{\varepsilon\}$ and $L^{n+1} = L.L^n$, where ε denotes the empty word.

The semantics of $\mathrm{PDL}^{(0)}$ comes from the semantics of modal logic. The structures over which programs and formulae of $\mathrm{PDL}^{(0)}$ are interpreted are called *Kripke structures*. A *Kripke structure*, also called a (Kripke) model, is a tuple $M = \langle W, \tau, (R_\sigma)_{\sigma\in\Pi_0}, h\rangle$, where W is a set of states, $\tau \in W$ is the current state, R_σ for $\sigma \in \Pi_0$ is a binary relation on W (representing the set of input/output pairs of states of the program σ), and h is a function mapping states to sets of atomic propositions. For $w \in W$, the set of atomic propositions "true" at w is $h(w)$.

Given a Kripke model $M = \langle W, \tau, (R_\sigma)_{\sigma\in\Pi_0}, h\rangle$, define $R_{\pi\cup\pi'} = R_\pi \cup R_{\pi'}$, $R_{\pi;\pi'} = R_\pi \circ R_{\pi'}$, $R_{\pi^*} = R_\pi^*$, where $R_\pi^* = \bigcup_{n\geq 0} R_\pi^n$ with $R_\pi^0 = \{(w,w) \mid w \in W\}$ and $R_\pi^{n+1} = R_\pi \circ R_\pi^n$. It is easily seen that for $\pi \in \Pi$, $R_\pi = \bigcup_{\alpha\in\mathcal{L}(\pi)} R_\alpha$.

Given a Kripke model $M = \langle W, \tau, (R_\sigma)_{\sigma\in\Pi_0}, h\rangle$, a state $w \in W$, and a formula φ, the satisfaction relation $M, w \models \varphi$ is defined as usual for the classical connectives and that:

$$
\begin{array}{lll}
M, w \models p & \text{iff} & p \in h(w) \\
M, w \models [\pi]\varphi & \text{iff} & \forall v \in W. R_\pi(w,v) \text{ implies } M, v \models \varphi \\
M, w \models \langle\pi\rangle\varphi & \text{iff} & \exists v \in W. R_\pi(w,v) \text{ and } M, v \models \varphi
\end{array}
$$

We say that M *satisfies* φ and φ is *true* in M, written $M \models \varphi$, if $M, \tau \models \varphi$. Let Γ be a formula set. We write $M \models \Gamma$ to denote that $M \models \varphi$ for every $\varphi \in \Gamma$. If $M \models \Gamma$ then we call M a *model* of Γ and say that Γ is *satisfiable*. We write $\Gamma \models \varphi$ to denote that every model of Γ satisfies φ.

2.2 The Deterministic Horn Fragment of $\mathrm{PDL}^{(0)}$

We extend the primitive language with universal modal operators $[\pi]_\diamond$, which have the same role as the modal operator $\circledast$ discussed in the Introduction. The semantics of $[\pi]_\diamond$ is defined as follows: $M, w \models [\pi]_\diamond\varphi$ iff $M, w \models [\pi]\varphi$ and for every $\alpha, \beta \in \Pi_0^*$, $\sigma \in \Pi_0$, if $\alpha\sigma\beta \in \mathcal{L}(\pi)$ then $M, w \models [\alpha]\langle\sigma\rangle\top$.

Note that $[\sigma]_\diamond\varphi \equiv [\sigma]\varphi \wedge \langle\sigma\rangle\top$ like the case of $\circledast$, but in general we do not have that $[\pi]_\diamond\varphi \equiv [\pi]\varphi \wedge \langle\pi\rangle\top$. Informally, $\langle\pi\rangle\top$ means that there exists a run of π (with the stop property), while the additional condition of $[\pi]_\diamond$

[3]The semantics of $\varphi?$ w.r.t. a Kripke model $M = \langle W, \tau, (R_\sigma)_{\sigma\in\Pi_0}, h\rangle$ is specified by $R_{\varphi?} = \{(w,w) \mid M, w \models \varphi\}$.

(w.r.t. $[\pi]$) means that every partial run of π is not blocked. Also note that formulae of the form $[\pi]_\diamond\varphi$ are expressible in PDL$^{(0)}$.[4]

A *positive formula* is a formula (in the extended language) without the connectives $\rightarrow$ and $\neg$. A *serial positive formula* is a positive formula which may contain modal operators $\langle\pi\rangle$ and $[\pi]_\diamond$ but not $[\pi]$ (and $[A]$ defined later for a finite automaton A).

A *deterministic Horn formula* in PDL$^{(0)}$ is a formula of one of the forms:

- $\top$ or an atomic proposition;
- $\neg\varphi$ or $\varphi \rightarrow \psi$, where φ is a serial positive formula and ψ is a deterministic Horn formula;
- $\varphi \wedge \psi$, where φ and ψ are deterministic Horn formulae;
- $[\pi]\varphi$ or $[\pi]_\diamond\varphi$, where φ is a deterministic Horn formula;
- $\langle\pi\rangle\varphi$, where φ is a deterministic Horn formula and π is a program without $\cup$ and *.

A *deterministic program clause* in PDL$^{(0)}$ is a formula of one of the forms:

$$B_1 \wedge \ldots \wedge B_n \rightarrow A$$
$$[\pi](B_1 \wedge \ldots \wedge B_n \rightarrow A)$$

and a *deterministic negative clause* in PDL$^{(0)}$ is a formula of one of the forms:

$$B_1 \wedge \ldots \wedge B_n \rightarrow \bot$$
$$[\pi](B_1 \wedge \ldots \wedge B_n \rightarrow \bot)$$

where $n \geq 0$, $B_1, \ldots, B_n$ are formulae of the form p, $[\alpha]_\diamond p$, $[\alpha]_\diamond\top$, $\langle\alpha\rangle p$, or $\langle\alpha\rangle\top$, A is a formula of the form p, $[\alpha]p$, $\langle\beta\rangle p$, or $\langle\beta\rangle\top$, where β is a program without $\cup$ and *, and $\bot$ denotes $\neg\top$.

A *deterministic Horn clause* in PDL$^{(0)}$ is either a deterministic program clause or a deterministic negative clause. (This notion is used for Proposition 1 and Corollary 2 given below.)

A *deterministic positive logic program* in PDL$^{(0)}$ is a finite set of deterministic program clauses.

PROPOSITION 1. *Every set of deterministic Horn formulae can be transformed into a set of deterministic Horn clauses preserving satisfiability.*

Sketch of the proof Apply the technique of [7], which is based on replacing a complex formula by a fresh atomic proposition and adding a formula defining that atomic proposition. For example, $[\pi]([\pi']_\diamond\varphi \rightarrow \psi)$, where φ is not $\top$ or an atomic proposition, is replaced by $[\pi]([\pi']_\diamond p \rightarrow \psi)$ and $[\pi;\pi'](\varphi \rightarrow p)$, where p is a fresh atom proposition. ■

[4]Let $A = \langle\Pi_0, Q, q_I, \delta, F\rangle$ be a deterministic finite automaton equivalent to π. Let $S \subseteq Q \times \Pi_0$ be the set of pairs (q,σ) such that $\delta(q,\sigma)$ is productive in A. For $(q,\sigma) \in S$, let $\pi_{q,\sigma}$ be the regular expression equivalent to the automaton $\langle\Pi_0, Q, q_I, \delta, \{q\}\rangle$. Then $[\pi]_\diamond\varphi$ is expressible in PDL$^{(0)}$ as $[\pi]\varphi \wedge \bigwedge_{(q,\sigma)\in S}[\pi_{q,\sigma}]\langle\sigma\rangle\top$.

COROLLARY 2. *Every finite set Γ of deterministic Horn formulae can be transformed into a deterministic positive logic program P and a serial positive formula φ such that Γ is unsatisfiable iff $P \models \varphi$.*

For the proof, just note that $[\pi](B_1 \wedge \ldots \wedge B_n \to \bot) \equiv \neg\langle\pi\rangle(B_1 \wedge \ldots \wedge B_n)$.

2.3 Automaton-Modal Operators

Recall that a *finite automaton* A is a tuple $\langle\Sigma, Q, I, \delta, F\rangle$, where Σ is the alphabet, Q is a finite set of states, $I \subseteq Q$ is the set of initial states, $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation, and $F \subseteq Q$ is the set of accepting states. A *run* of A on a word $\sigma_1 \ldots \sigma_k$ is a finite sequence of states $q_0, q_1, \ldots, q_k$ such that $q_0 \in I$ and $\delta(q_{i-1}, \sigma_i, q_i)$ holds for every $1 \leq i \leq k$. It is an *accepting run* if $q_k \in F$. We say that A *accepts* word w if there exists an accepting run of A on w. The set of all words accepted/recognized by A is denoted by $\mathcal{L}(A)$.

If A is a finite automaton then we call $[A]$ a (universal) *automaton-modal operator*. If A is a finite automaton and φ is a formula in the primitive language, i.e. $\varphi \in \Phi$, then we call $[A]\varphi$ a formula in the extended language.

The semantics of formulae with automaton-modal operators are defined as follows: $M, w_0 \models [A]\varphi$ if $M, w_k \models \varphi$ for every path $w_0 R_{\sigma_1} w_1 \ldots w_{k-1} R_{\sigma_k} w_k$ with $k \geq 0$ and $\sigma_1 \ldots \sigma_k \in \mathcal{L}(A)$.

It is well known that every regular expression π is equivalent to a finite automaton A (with the same alphabet) in the sense that $\mathcal{L}(\pi) = \mathcal{L}(A)$. It is easy to see that if π is equivalent to A then $M, w \models [\pi]\varphi$ iff $M, w \models [A]\varphi$. For every regular expression π, let $A_\pi = \langle\Pi_0, Q_\pi, I_\pi, \delta_\pi, F_\pi\rangle$ be a fixed finite automaton equivalent to π. A_π can be constructed from π in linear time. We assume that every state of Q_π is reachable from some state of I_π via a path using the transition relation δ_π.

If A is a finite automaton and Q is a subset of the states of A, then by (A, Q) we denote the finite automaton obtained from A by using Q as the set of initial states, and we will write $[A, Q]$ for the automaton-modal operator $[(A, Q)]$.

For a finite automaton $A = \langle\Pi_0, Q_A, I_A, \delta_A, F_A\rangle$ and $\alpha \in \Pi_0^*$, define

$$\begin{array}{rcl} \delta_A(Q,\sigma) & = & \{q' \mid \exists q \in Q.(q,\sigma,q') \in \delta_A\}, \\ \widetilde{\delta_A}(Q,\varepsilon) & = & Q, \\ \widetilde{\delta_A}(Q,\alpha\sigma) & = & \delta_A(\widetilde{\delta_A}(Q,\alpha),\sigma). \end{array}$$

We have that $M, w_0 \models [A, Q]\varphi$ iff $M, w_k \models \varphi$ for every path $w_0 R_{\sigma_1} w_1 \ldots w_{k-1} R_{\sigma_k} w_k$ with $k \geq 0$ and $\widetilde{\delta_A}(Q, \sigma_1 \ldots \sigma_k) \cap F_A \neq \emptyset$.

2.4 Pseudo-models for Dealing with Non-seriality

A *pseudo-model* is a tuple $M = \langle W, \tau, (R_\sigma)_{\sigma\in\Pi_0}, (S_\sigma)_{\sigma\in\Pi_0}, h\rangle$, which is similar to a model except that for every $\sigma \in \Pi_0$, there are two accessibility relations R_σ and S_σ. We require that $R_\sigma \subseteq S_\sigma$ for every $\sigma \in \Pi_0$. The accessibility relations R_σ, resp. S_σ, are used to deal with existential, resp. universal, modal operators.

Given a pseudo-model $M = \langle W, \tau, (R_\sigma)_{\sigma\in\Pi_0}, (S_\sigma)_{\sigma\in\Pi_0}, h\rangle$, for $\pi \in \Pi$, define S_π analogously as for R_π; and for $w \in W$, define the relation $M, w \models \varphi$ in the usual way for classical connectives and:

- $M, w \models \langle\pi\rangle\varphi$ iff there exists $v \in W$ such that $R_\pi(w, v)$ and $M, v \models \varphi$;
- $M, w \models [\pi]\varphi$ iff for every $v \in W$, $S_\pi(w, v)$ implies $M, v \models \varphi$;
- $M, w \models [\pi]_\diamond\varphi$ iff $M, w \models [\pi]\varphi$ and for every $\alpha, \beta \in \Pi_0^*$, $\sigma \in \Pi_0$, if $\alpha\sigma\beta \in \mathcal{L}(\pi)$ then $M, w \models [\alpha]\langle\sigma\rangle\top$;
- $M, w \models [A]\varphi$ iff $M, w_k \models \varphi$ for every path $w_0 S_{\sigma_1} w_1 \ldots w_{k-1} S_{\sigma_k} w_k$ with $k \geq 0$ and $\sigma_1 \ldots \sigma_k \in \mathcal{L}(A)$.

Other related definitions remain unchanged.

If $M \models \Gamma$ then we say that M is a pseudo-model of Γ.

Every model is also a pseudo-model, with $S_\sigma = R_\sigma$ for every $\sigma \in \Pi_0$.

PROPOSITION 3. *The problem of checking $M \models \varphi$ for a given pseudo-model M and a given formula φ is solvable in polynomial time in the sizes of M and φ.*

This proposition can be proved by induction on the construction of φ. To deal with modal operators, we can run corresponding automata along paths in M.

2.5 Ordering Pseudo-models

In [7] we introduced an ordering between Kripke models. In this subsection, we provide an analogue for ordering pseudo-models. A pseudo-model M is said to be *less than* or *equal to* a pseudo-model M', write $M \leq M'$, if for every positive formula φ (in the extended language with $[\pi]_\diamond$ and $[A]$), if $M \models \varphi$ then $M' \models \varphi$. This relation $\leq$ is a pre-order.[5] We write $M \equiv M'$ to denote that $M \leq M'$ and $M' \leq M$.

A pseudo-model M is a *least pseudo-model* of a deterministic positive logic program P if $M \models P$ and $M \leq M'$ for every pseudo-model M' of P.

Let $M = \langle W, \tau, (R_\sigma)_{\sigma\in\Pi_0}, (S_\sigma)_{\sigma\in\Pi_0}, h\rangle$ and $M' = \langle W', \tau', (R'_\sigma)_{\sigma\in\Pi_0}, (S'_\sigma)_{\sigma\in\Pi_0}, h'\rangle$ be pseudo-models. We say that M is *less than or equal to M' w.r.t. a binary relation $r \subseteq W \times W'$*, and write $M \leq_r M'$, if the following conditions hold:

1. $r(\tau, \tau')$
2. $\forall\,\sigma \in \Pi_0\ \forall\, x, x', y\ \ R_\sigma(x, y) \wedge r(x, x') \to \exists y'\ R'_\sigma(x', y') \wedge r(y, y')$
3. $\forall\,\sigma \in \Pi_0\ \forall\, x, x', y'\ \ S'_\sigma(x', y') \wedge r(x, x') \to \exists y\ S_\sigma(x, y) \wedge r(y, y')$
4. $\forall\, x, x'\ \ r(x, x') \to h(x) \subseteq h(x')$

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

In the above definition, the first three conditions state that r is a forward-backward bisimulation of the frames of M and M'.[6] Intuitively, $r(x, x')$ states that the state x is less than or equal to x'.

LEMMA 4. *If $M \leq_r M'$ then $M \leq M'$.*

Proof. Let $M = \langle W, \tau, (R_\sigma)_{\sigma\in\Pi_0}, (S_\sigma)_{\sigma\in\Pi_0}, h\rangle$ and $M' = \langle W', \tau', (R'_\sigma)_{\sigma\in\Pi_0}, (S'_\sigma)_{\sigma\in\Pi_0}, h'\rangle$ be pseudo-models and suppose that $M \leq_r M'$. We prove that for every positive formula φ and every u, u' such that $r(u, u')$, if $M, u \models \varphi$ then $M', u' \models \varphi$. We do this by induction on the construction of φ. Suppose that $r(u, u')$ holds and $M, u \models \varphi$.

The cases when $\varphi = p$ or $\varphi = \psi \wedge \zeta$ or $\varphi = \psi \vee \zeta$ are trivial.

Case $\varphi = \langle\pi\rangle\psi$: Since $M, u \models \langle\pi\rangle\psi$, there exists $v \in W$ such that $R_\pi(u, v)$ holds and $M, v \models \psi$. There exist $\sigma_1, \ldots, \sigma_k \in \Pi_0$ such that $\sigma_1 \ldots \sigma_k \in \mathcal{L}(\pi)$ and $R_{\sigma_1;\ldots;\sigma_k}(u, v)$ holds. Let $u_0 = u$, u_1, …, u_{k-1}, $u_k = v$ be states such that $R_{\sigma_i}(u_{i-1}, u_i)$ holds for $1 \leq i \leq k$. Let $u'_0 = u'$. For every $1 \leq i \leq k$, by the condition 2, there exists a state $u'_i \in W'$ such that $R'_{\sigma_i}(u'_{i-1}, u'_i)$ and $r(u_i, u'_i)$ hold. Hence, $R'_\pi(u', v')$ and $r(v, v')$ hold for $v = u'_k$. Since $r(v, v')$ holds and $M, v \models \psi$, by the inductive assumption, $M', v' \models \psi$. Hence $M', u' \models \langle\pi\rangle\psi$.

Case $\varphi = [\pi]\psi$: Let v' be an arbitrary state of W' such that $S'_\pi(u', v')$ holds. There exist $\sigma_1, \ldots, \sigma_k \in \Pi_0$ such that $\sigma_1 \ldots \sigma_k \in \mathcal{L}(\pi)$ and $S'_{\sigma_1;\ldots;\sigma_k}(u', v')$ holds. Let $u'_0 = u'$, u'_1, …, u'_{k-1}, $u'_k = v'$ be states such that $S'_{\sigma_i}(u'_{i-1}, u'_i)$ holds for $1 \leq i \leq k$. Let $u_0 = u$. For every $1 \leq i \leq k$, by the condition 3, there exists a state $u_i \in W$ such that $S_{\sigma_i}(u_{i-1}, u_i)$ and $r(u_i, u'_i)$ hold. Hence, $S_\pi(u, v)$ and $r(v, v')$ hold for $v = u_k$. Since $S_\pi(u, v)$ holds and $M, u \models [\pi]\psi$, we have that $M, v \models \psi$. Since $r(v, v')$ holds and $M, v \models \psi$, by the inductive assumption, $M', v' \models \psi$. Hence $M', u' \models [\pi]\psi$.

The case $\varphi = [A]\psi$ is similar to the case $\varphi = [\pi]\psi$.

The proof for the case $\varphi = [\pi]_\diamond\psi$ is a combination of the proofs of the case $\varphi = [\pi]\psi$ and the case $\varphi = \langle\pi\rangle\psi$. ■

3 Constructing Finite Least Pseudo-models

In this section, we present an algorithm that, given a deterministic positive logic program P in PDL$^{(0)}$, constructs a finite least pseudo-model of P.

Let X be a set of formulae, which may contain modal operators of the form $[\pi]_\diamond$ or $[A, Q]$. The saturation of X, denoted by $\mathsf{Sat}(X)$, is defined to be the least extension of X such that:

- $\top \in \mathsf{Sat}(X)$,
- if $\langle\pi;\pi'\rangle\varphi \in \mathsf{Sat}(X)$ then $\langle\pi\rangle\langle\pi'\rangle\varphi \in \mathsf{Sat}(X)$,
- if $[\pi]\varphi \in \mathsf{Sat}(X)$ then $[A_\pi, I_\pi]\varphi \in \mathsf{Sat}(X)$,
- if $[A_\pi, Q]\varphi \in \mathsf{Sat}(X)$ and $Q \cap F_\pi \neq \emptyset$ then $\varphi \in \mathsf{Sat}(X)$.

[6] The condition 2 corresponds to the forward direction, while the condition 3 corresponds to the backward direction.

The transfer of X through $\langle\sigma\rangle$, where $\sigma \in \Pi_0$, is defined as follows:

$$\mathsf{Trans}(X, \sigma) = \mathsf{Sat}(\{[A_\pi, \delta_\pi(Q, \sigma)]\varphi \mid [A_\pi, Q]\varphi \in X\}).$$

The compact form $\mathsf{CF}(X)$ of X is the least set of formulae obtained as follows:

- if $\varphi \in X$ and φ is not of the form $[A_\pi, Q]\varphi$ then $\varphi \in \mathsf{CF}(X)$,
- if $[A_\pi, Q]\varphi \in X$ and $Q_1, \ldots, Q_k$ are all the sets such that $[A_\pi, Q_i]\varphi \in X$ for $1 \leq i \leq k$, then $[A_\pi, Q_1 \cup \ldots \cup Q_k]\varphi \in \mathsf{CF}(X)$.

We use the following data structures:

- W : a set of states, where $\tau \in W$ is the current state.
- H : for every $w \in W$, $H(w)$ is a set of formulae called the content of w.
- $Next : W \times \{\langle\sigma\rangle\varphi \mid \sigma \in \Pi_0,\ \varphi \in \Phi\} \to W$, a partial function interpreted as follows: $Next(u, \langle\sigma\rangle\varphi) = v$ means $\langle\sigma\rangle\varphi \in H(u)$, $\varphi \in H(v)$, and $\langle\sigma\rangle\varphi$ is "realized" at u by going to v via R_σ.
- $Next_S : W \times \Pi_0 \to W$, where $Next_S(u, \sigma) = v$ implies $S_\sigma(u, v)$.

Using the above data structures, we define:

- h to be the restriction of H such that $h(u) = H(u) \cap \Phi_0$ for $u \in W$;
- R_σ, for $\sigma \in \Pi_0$, to be $\{(u, v) \mid Next(u, \langle\sigma\rangle\varphi) = v$ for some $\varphi\}$;
- S_σ, for $\sigma \in \Pi_0$, to be $R_\sigma \cup \{(u, v) \mid Next_S(u, \sigma) = v\}$;
- $M = \langle W, \tau, (R_\sigma)_{\sigma\in\Pi_0}, (S_\sigma)_{\sigma\in\Pi_0}, h\rangle$.

In the algorithm given below, we use the function $\mathsf{Find}(X)$ defined as follows: if there exists a state $u \in W$ with $H(u) = X$ then return u, else add a new state u to W with $H(u) = X$ and return u.

A pseudo-model of P is constructed by building a "pseudo-model graph" for P. At the beginning the pseudo-model graph contains only one state with content P. Then for every state u and every formula φ belonging to the content of u, if φ is not true at u then the algorithm makes a change to satisfy it. There are three main forms for φ: $[A_\pi, Q]\psi$, $(B_1 \wedge \ldots \wedge B_k \to A)$, and $\langle\sigma\rangle\psi$ (the form $[\pi]\psi$ is reduced to $[A_\pi, I_\pi]\psi$, and the form $\langle\pi\rangle\psi$ is reduced to $\langle\sigma\rangle\psi'$). For the case when φ is of the form $[A_\pi, Q]\psi$, for every $\sigma \in \Pi_0$, we would like to add $[A_\pi, \delta_\pi(Q, \sigma)]\psi$ to the content of every state w accessible from u via S_σ. But such an action may affect other states involved with w. So, instead of adding the formula to the content of w, we discard the connection $S_\sigma(u, w)$ and connect u via S_σ (in an appropriate way using $Next$ or $Next_S$) to a state w_* with an appropriate content, which is created if necessary. That is we use w_* to replace the role of w. For the case of $(B_1 \wedge \ldots \wedge B_k \to A)$, if all $B_1, \ldots, B_k$ are "certainly" true at u (the truth of $[\pi]_\diamond p$ at u is checked in a special way) then we would like to add A to the content of u. But analogously as for the previous case, instead of modifying the content of u, we just redirect connections appropriately. States are cached and never deleted. For the case when φ is of the form $\langle\sigma\rangle\psi$, to satisfy φ at u, we connect u via R_σ to the state with content consisting of ψ and the formulae "inherited" from u via R_σ. To guarantee the constructed pseudo-model to be smallest, for every $u \in W$ and $\sigma \in \Pi_0$, we connect u via S_σ using $Next_S$ to the state with content inherited from u via S_σ. Such connections are also useful for checking the truth of formulae of the form $[\pi]_\diamond p$ in a state.

ALGORITHM 5.
Input: A deterministic positive logic program P in PDL$^{(0)}$.
Output: $M = \langle W, \tau, (R_\sigma)_{\sigma\in\Pi_0}, (S_\sigma)_{\sigma\in\Pi_0}, h\rangle$:
a finite least pseudo-model of P.

1. $W := \{\tau\}$; $H(\tau) := \mathsf{CF}(\mathsf{Sat}(P))$;
2. for every $u \in W$ and every $\varphi \in H(u)$
   (a) case $\varphi = [A_\pi, Q]\psi$:
   for every $w \in W$ and $\sigma \in \Pi_0$ such that $S_\sigma(u, w)$ holds:
      i. $w_* := \mathsf{Find}(\mathsf{CF}(H(w) \cup \mathsf{Trans}(\{\varphi\}, \sigma)))$;
      ii. for every $\xi \in \Phi$,
      if $Next(u, \langle\sigma\rangle\xi) = w$ then $Next(u, \langle\sigma\rangle\xi) := w_*$;
      iii. if $Next_S(u, \sigma) = w$ then $Next_S(u, \sigma) := w_*$;
   (b) case $\varphi = (B_1 \wedge \ldots \wedge B_k \to A)$:
   if
   for every $1 \leq i \leq k$, $M, u \models B_i$ and if $B_i = [\pi]_\diamond p$ then for every $w \in W$ and $\sigma \in \Pi_0$, if $S_\alpha(u, w)$ holds and $\alpha\sigma\beta \in \mathcal{L}(\pi)$ for some $\alpha, \beta \in \Pi_0^*$, then $Next_S(w, \sigma)$ is defined,
   then
      i. $u_* := \mathsf{Find}(\mathsf{CF}(H(u) \cup \mathsf{Sat}(\{A\})))$;
      ii. for every $v \in W$, $\sigma \in \Pi_0$, and $\psi \in \Phi$,
      if $Next(v, \langle\sigma\rangle\psi) = u$ then $Next(v, \langle\sigma\rangle\psi) := u_*$;
      iii. for every $v \in W$ and $\sigma \in \Pi_0$,
      if $Next_S(v, \sigma) = u$ then $Next_S(v, \sigma) := u_*$;
      iv. if $\tau = u$ then $\tau := u_*$;
   (c) case $\varphi = \langle\sigma\rangle\psi$:
   if $Next(u, \langle\sigma\rangle\psi)$ is not defined then
   $Next(u, \langle\sigma\rangle\psi) := \mathsf{Find}(\mathsf{CF}(\mathsf{Trans}(H(u), \sigma) \cup \mathsf{Sat}(\{\psi\})))$;
3. for every $u \in W$ and every $\sigma \in \Pi_0$,
   if $Next_S(u, \sigma)$ is not defined then
   $Next_S(u, \sigma) := \mathsf{Find}(\mathsf{CF}(\mathsf{Trans}(H(u), \sigma)))$;

4. while some change occurred, go to step 2.

PROPOSITION 6. *Algorithm 5 terminates in $2^{O(n^2)}$ steps and returns a pseudo-model with $2^{O(n^2)}$ states, where n is the size of P (i.e. the sum of the lengths of the clauses of P).*

Proof. For each $u \in W$ and $\varphi \in H(u)$, there are three cases: i) φ is a subformula of a clause of P, ii) φ is a formula of the form $[A_\pi, Q]\psi$ with $[\pi]\psi$ being a subformula of a clause of P, iii) φ is a formula of the form $\langle\pi\rangle\psi$, where ψ is a subformula of a clause of P and π is a subprogram occurring in P. There are less than n possible values for ψ and π, and less than 2^n possible values for Q. Hence, due to the compact form, there are no more than $2^{O(n^2)}$ possible values for $H(u)$. Since the states of W have different contents, the size of W is $2^{O(n^2)}$.

The if condition of the step 2b can be checked in $2^{O(n^2)}$ steps.

The steps 2c and 3 make a change no more than $2^{O(n^2)}.n.n = 2^{O(n^2)}$ times. For the steps 2a and 2b, note that the content of u_* (resp. w_*) is "bigger" than the content of u (resp. w). Hence $Next$ and $Next_S$ are modified by the steps 2a or 2b no more than $2^{O(n^2)}.n.n.2^{O(n^2)} = 2^{O(n^2)}$ times, and τ is modified no more than $2^{O(n^2)}$ times.

Therefore, Algorithm 5 terminates in $2^{O(n^2)}$ steps and returns a pseudo-model with $2^{O(n^2)}$ states, where each state is of size $O(n)$. ∎

LEMMA 7. *Let P be a deterministic positive logic program in $PDL^{(0)}$ and M the pseudo-model constructed by Algorithm 5 for P. Then $M \models P$.*

Proof. We will refer to the data structures used in Algorithm 5.

For $u, u' \in W$, we write $H(u) \leq H(u')$ to denote that, for every $\varphi \in H(u)$, either $\varphi \in H(u')$ or $\varphi = [A_\pi, Q]\psi$ and there exists $[A_\pi, Q']\psi \in H(u')$ with $Q' \supseteq Q$. Observe that, for every v, σ, ψ, if $Next(v, \langle\sigma\rangle\psi)$ or $Next_S(v, \sigma)$ changes its current value from u to u' then $H(u) \leq H(u')$.

To prove that $M \models P$, we show that for every $u \in W$ reachable from τ via a path using the accessibility relations $(S_\sigma)_{\sigma\in\Pi_0}$ and for every formula $\varphi \in H(u)$ without automaton-modal operators, $M, u \models \varphi$. We prove this by induction on the structure of φ.

Consider the case when $\varphi = [\pi]\psi$. Suppose that $S_\pi(u, w)$ holds. By the inductive assumption, it is sufficient to show that $\psi \in H(w)$. There exist $w_0, \ldots, w_k$ in W with $w_0 = u$, $w_k = w$, and $\sigma_1, \ldots, \sigma_k \in \Pi_0$ such that $\sigma_1 \ldots \sigma_k \in \mathcal{L}(\pi)$ and $S_{\sigma_i}(w_{i-1}, w_i)$ holds for $1 \leq i \leq k$. Since $[\pi]\psi \in H(w_0)$, we have $[A_\pi, Q]\psi \in H(w_0)$ for some $Q \supseteq I_\pi$. Hence, for every $1 \leq i \leq k$, there exists $[A_\pi, Q_i]\psi \in H(w_i)$ with $Q_i \supseteq \widetilde{\delta_\pi}(I_\pi, \sigma_1 \ldots \sigma_i)$. Since $\sigma_1 \ldots \sigma_k \in \mathcal{L}(\pi)$, $\widetilde{\delta_\pi}(I_\pi, \sigma_1 \ldots \sigma_k) \cap F_\pi \neq \emptyset$, and hence $Q_k \cap F_\pi \neq \emptyset$ and $\psi \in H(w)$.

Consider the case when $\varphi = (B_1 \wedge \ldots \wedge B_k \rightarrow A)$ and the steps 2(b)ii, 2(b)iii, 2(b)iv are executed. As no changes occur (at the end) and u is reachable from τ via a path using the accessibility relations $(S_\sigma)_{\sigma\in\Pi_0}$, we have that $u_* = u$. Thus, by the inductive assumption, $M, u \models A$, and hence $M, u \models \varphi$.

The case $\varphi = \langle\pi\rangle\psi$ is reduced to the case $\varphi = \langle\sigma\rangle\psi'$, which is trivial. ■

LEMMA 8. *Let P be a deterministic positive logic program in $PDL^{(0)}$ and $M' = \langle W', \tau', (R'_\sigma)_{\sigma\in\Pi_0}, (S'_\sigma)_{\sigma\in\Pi_0}, h'\rangle$ be an arbitrary pseudo-model of P. Consider a moment after executing a numerated step in an execution of Algorithm 5 for P. Let $r = \{(x, x') \in W \times W' \mid M', x' \models H(x)\}$. Then the following conditions hold:*

- $r(\tau, \tau')$
- $\forall x, y, x', y', \sigma, \psi$
  $$r(x, x') \wedge (Next(x, \langle\sigma\rangle\psi) = y) \wedge R'_\sigma(x', y') \wedge (M', y' \models \psi) \to r(y, y')$$
- $\forall x, y, x', y', \sigma \;\; r(x, x') \wedge (Next_S(x, \sigma) = y) \wedge S'_\sigma(x', y') \to r(y, y')$

Proof. By induction on the number of executed steps.

The base case occurs after executing step 1 and the assertions clearly hold. Consider some latter step of the algorithm. As induction hypothesis, assume that the assertions hold before executing that step. Suppose that after executing the step we have r_2, W_2, H_2, $Next_2$, $Next_{S2}$, $R_{2,\sigma}$, $S_{2,\sigma}$ (for $\sigma \in \Pi_0$), and M_2 in the places of r, W, H, $Next$, $Next_S$, R_σ, S_σ, and M. We prove that:

- $r_2(\tau, \tau')$
- $\forall x, y, x', y', \sigma, \psi$
  $$r_2(x, x') \wedge (Next_2(x, \langle\sigma\rangle\psi) = y) \wedge R'_\sigma(x', y') \wedge (M', y' \models \psi) \to r_2(y, y')$$
- $\forall x, y, x', y', \sigma \;\; r_2(x, x') \wedge (Next_{S2}(x, \sigma) = y) \wedge S'_\sigma(x', y') \to r_2(y, y')$

It suffices to consider steps 2(a)ii, 2(a)iii, 2(b)ii-2(b)iv, 2c, and 3.

Consider the step 2(a)ii. It suffices to show that if $r(u, u') \wedge (Next(u, \langle\sigma\rangle\psi) = w) \wedge R'_\sigma(u', w') \wedge (M', w' \models \psi)$ then $M', w' \models H(w_*)$. Suppose that the premise holds. By the inductive assumption, $r(w, w')$ holds and $M', w' \models H(w)$. Since $r(u, u')$ holds and $[A_\pi, Q]\psi \in H(u)$, we have that $M', u' \models [A_\pi, Q]\psi$. Hence $M', w' \models [A_\pi, \delta_\pi(Q, \sigma)]\psi$ (since $R'_\sigma(u', w')$ holds). Hence $M', w' \models H(w_*)$.

Consider the step 2(a)iii. It suffices to show that if $r(u, u') \wedge (Next_S(u, \sigma) = w) \wedge S'_\sigma(u', w')$ holds then $M', w' \models H(w_*)$. This can be proved analogously as for the step 2(a)ii.

Consider the steps 2(b)ii-2(b)iv. Let u' be a state of W' such that $r(u, u')$ holds. It is sufficient to show that $r_2(u_*, u')$ holds. We need only to show that $M', u' \models A$. Since $r(u, u')$ holds, $M', u' \models (B_1 \wedge \ldots \wedge B_k \to A)$. Hence, it is sufficient to show that $M', u' \models B_i$ for every $1 \leq i \leq k$. Fix such an index i. There are three cases to consider:

- Case $B_i = p$: Since $M, u \models B_i$, we have that $p \in H(u)$. Since $r(u, u')$ holds, it follows that $M', u' \models B_i$.

- Case $B_i = \langle\pi\rangle p$: There exists $w \in W$ such that $R_\pi(u,w)$ holds and $p \in H(w)$. Thus, there exist $w_0, \ldots, w_k$ in W with $w_0 = u$, $w_k = w$, and $\sigma_1, \ldots, \sigma_k \in \Pi_0$ such that $\sigma_1 \ldots \sigma_k \in \mathcal{L}(\pi)$ and $R_{\sigma_i}(w_{i-1}, w_i)$ holds for $1 \leq i \leq k$. Let $\psi_1, \ldots, \psi_k$ be formulae such that $Next(w_{i-1}, \langle\sigma_i\rangle\psi_i) = w_i$ for $1 \leq i \leq k$. Let $w'_0 = u$. Since $r(w_0, w'_0)$ holds and $\langle\sigma_1\rangle\psi_1 \in H(w_0)$, we have that $M', w'_0 \models \langle\sigma_1\rangle\psi_1$. Hence, there exists $w'_1 \in W'$ such that $R'_{\sigma_1}(w'_0, w'_1)$ holds and $M', w'_1 \models \psi_1$. By the inductive assumption, $r(w_1, w'_1)$ holds. Analogously, there exist $w'_2, \ldots, w'_k \in W'$ such that $R'_{\sigma_i}(w'_{i-1}, w'_i)$ and $r(w_i, w'_i)$ hold for every $1 \leq i \leq k$. Thus $R'_\pi(w'_0, w'_k)$ holds. Since $p \in H(w)$, $w = w_k$, and $r(w_k, w'_k)$ holds, we have $M', w'_k \models p$. It follows that $M', w'_0 \models \langle\pi\rangle p$, which means $M', u' \models B_i$.

- Case $B_i = [\pi]_\diamond p$:
  - We first show that $M', u' \models [\pi]p$. Suppose that $S'_\pi(u', w')$ holds. There exist $\sigma_1, \ldots, \sigma_k \in \Pi_0$ and $w'_0 = u', w'_1, \ldots, w'_k \in W'$ such that $\sigma_1 \ldots \sigma_k \in \mathcal{L}(\pi)$ and $S'_{\sigma_i}(w'_{i-1}, w'_i)$ holds for every $1 \leq i \leq k$. Let $w_0 = u$ and $w_i = Next_S(w_{i-1}, \sigma_i)$ for $1 \leq i \leq k$. The "if" condition of step 2b guarantees the existence of the states w_i. By the inductive assumption, $r(w_i, w'_i)$ holds for $1 \leq i \leq k$. Since $M, u \models [\pi]_\diamond p$, we have that $p \in H(w_k)$. Since $r(w_k, w'_k)$ holds, it follows that $M', w'_k \models p$, which means $M', w' \models p$. This implies that $M', u' \models [\pi]p$.
  - Suppose that $\alpha\sigma\beta \in \mathcal{L}(\pi)$ and $S'_\alpha(u', w')$ holds, where $\alpha, \beta \in \Pi_0^*$ and $\sigma \in \Pi_0$. We show that $M', w' \models \langle\sigma\rangle\top$. Let $\alpha = \sigma_1 \ldots \sigma_k$. There exist $w'_0, \ldots, w'_k$ such that $w'_0 = u'$, $w'_k = w'$, $S'_{\sigma_i}(w'_{i-1}, w'_i)$ holds for every $1 \leq i \leq k$. Let $w_0 = u$ and $w_i = Next_S(w_{i-1}, \sigma_i)$ for $1 \leq i \leq k$. The "if" condition of step 2b guarantees the existence of the states w_i. By the inductive assumption, $r(w_i, w'_i)$ holds for $1 \leq i \leq k$. Since $M, u \models [\pi]_\diamond p$, we have that $M, w_k \models \langle\sigma\rangle\top$. Hence there exists $v \in W$ such that $R_\sigma(w_k, v)$ holds. Thus $v = Next(w_k, \langle\sigma\rangle\zeta)$ for some ζ, and $\langle\sigma\rangle\zeta \in H(w_k)$. Since $r(w_k, w'_k)$ holds, by the definition of r, $M', w'_k \models \langle\sigma\rangle\zeta$. Hence $M', w' \models \langle\sigma\rangle\top$.

Consider the step 2c. Let w denote the state $\mathsf{Find}(\mathsf{CF}(\mathsf{Trans}(H(u), \sigma) \cup \mathsf{Sat}(\{\psi\})))$. Suppose that $r(u, u')$ and $R'_\sigma(u', w')$ hold and $M', w' \models \psi$. It suffices to show that $M', w' \models H_2(w)$. Since $r(u, u')$ holds, $M', u' \models H(u)$. It follows that $M', w' \models \mathsf{Trans}(H(u), \sigma)$ (since $R'_\sigma(u', w')$ holds). Hence $M', w' \models H_2(w)$.

Consider the step 3. Let w denote the state $\mathsf{Find}(\mathsf{CF}(\mathsf{Trans}(H(u), \sigma)))$. Suppose that $r(u, u')$ and $S'_\sigma(u', w')$ hold. It suffices to show that $M', w' \models H_2(w)$. Since $r(u, u')$ holds, $M', u' \models H(u)$. It follows that $M', w' \models \mathsf{Trans}(H(u), \sigma)$ (since $S'_\sigma(u', w')$ holds). Hence $M', w' \models H_2(w)$. ■

LEMMA 9. *Let P be a deterministic positive logic program in $PDL^{(0)}$, M be the pseudo-model constructed by Algorithm 5 for P, and $M' = \langle W', \tau',$*

$(R'_\sigma)_{\sigma\in\Pi_0}$, $(S'_\sigma)_{\sigma\in\Pi_0}$, $h'\rangle$ *be an arbitrary pseudo-model of* P*. Then* $M \leq M'$*. In particular, if* r *is the relation defined as in Lemma 8, then* $M \leq_r M'$*.*

Proof. We will refer to the data structures used in Algorithm 5. Let r be the relation specified in Lemma 8 for the end of an execution of Algorithm 5 for P. By definition, $\forall\, x, x'\; r(x, x') \to h(x) \subseteq h(x')$ is true. By Lemma 8, $r(\tau, \tau')$ holds.

We prove that $\forall \sigma, x, x', y\; R_\sigma(x, y) \wedge r(x, x') \to \exists y'\; R'_\sigma(x', y') \wedge r(y, y')$. Suppose that $R_\sigma(x, y)$ and $r(x, x')$ hold. There exists ζ such that $y = Next(x, \langle\sigma\rangle\zeta)$. We have that $\langle\sigma\rangle\zeta \in H(x)$. Since $r(x, x')$ holds, $M', x' \models \langle\sigma\rangle\zeta$. There thus exists $y' \in W'$ such that $R'_\sigma(x', y')$ holds and $M', y' \models \zeta$. By Lemma 8, $r(y, y')$ holds.

We now prove that $\forall \sigma, x, x', y'\; S'_\sigma(x', y') \wedge r(x, x') \to \exists y\; S_\sigma(x, y) \wedge r(y, y')$. Suppose that $S'_\sigma(x', y')$ and $r(x, x')$ hold. Let $y = Next_S(x, \sigma)$. By Lemma 8, $r(y, y')$ holds.

We have prove that $M \leq_r M'$. Therefore $M \leq M'$. ■

THEOREM 10. *Let* P *be a deterministic positive logic program in PDL*$^{(0)}$*. The pseudo-model* M *constructed by Algorithm 5 for* P *is a least pseudo-model of* P*.*

This theorem follows from Lemmas 7 and 9.

4 Characterizations of Least Pseudo-models

In classical propositional logic, if M is the least model of a positive logic program P then for every positive formula φ, $P \models \varphi$ iff $M \models \varphi$. Similarly, in a basic serial monomodal logic L, if M is a least L-model of a positive modal logic program P then for every positive (modal) formula φ, $P \models_L \varphi$ iff $M \models \varphi$ (see [7]). In this section, we extend such an assertion for the deterministic Horn fragment of PDL$^{(0)}$. The main result says that if P is a deterministic positive logic program in PDL$^{(0)}$, M is a least pseudo-model of P, and φ is a serial positive formula, then $P \models \varphi$ iff $M \models \varphi$.

Given a pseudo-model $M = \langle W, \tau, (R_\sigma)_{\sigma\in\Pi_0}, (S_\sigma)_{\sigma\in\Pi_0}, h\rangle$, let $M' = \langle W$, τ, $(R'_\sigma)_{\sigma\in\Pi_0}$, $(S'_\sigma)_{\sigma\in\Pi_0}$, $h\rangle$ be the pseudo-model specified as follows:

$$\begin{aligned} R'_\sigma &= R_\sigma \cup \{(u, w) \mid S_\sigma(u, w) \text{ and } R_\sigma(u, w') \text{ hold for some } w'\}, \\ S'_\sigma &= S_\sigma \setminus \{(u, w) \mid w \in W \text{ and } R_\sigma(u, w') \text{ does not hold for any } w'\}. \end{aligned}$$

Thus, $R'_\sigma = S'_\sigma$ for every $\sigma \in \Pi_0$, and M' can be treated as a Kripke model. We call M' the *model corresponding to* M.

LEMMA 11. *Let* P *be a deterministic positive logic program in PDL*$^{(0)}$*,* M *the pseudo-model constructed by Algorithm 5 for* P*,* M' *the model corresponding to* M*, and* φ *a serial positive formula. If* $M' \models \varphi$ *then* $M \models \varphi$*.*

Proof. We will refer to the data structures used by Algorithm 5 for P and M. Let $r = \{(x, x') \in W \times W \mid M, x' \models H(x)\}$. By Lemma 8, we have that:

(i) $r(\tau, \tau)$

(ii) $\forall x, y, x', y', \sigma, \psi$
$$r(x, x') \wedge (Next(x, \langle\sigma\rangle\psi) = y) \wedge R_\sigma(x', y') \wedge (M, y' \models \psi) \to r(y, y')$$

(iii) $\forall x, y, x', y', \sigma \;\; r(x, x') \wedge (Next_S(x, \sigma) = y) \wedge S_\sigma(x', y') \to r(y, y')$

and by Lemma 9, $M \leq_r M$.

Let $M' = \langle W,\, \tau,\, (R'_\sigma)_{\sigma\in\Pi_0},\, (S'_\sigma)_{\sigma\in\Pi_0},\, h\rangle$. It suffices to prove by induction on the construction of φ that, for every $x, x' \in W$, if $r(x, x')$ holds and $M', x \models \varphi$ then $M, x' \models \varphi$. Suppose that $r(x, x')$ holds and $M', x \models \varphi$.

- Case $\varphi = p$: Since $M', x \models \varphi$, we have that $p \in h(x)$. Since $M \leq_r M$ and $r(x, x')$ holds, we derive that $p \in h(x')$. Hence $M, x' \models p$.

- Case $\varphi = \psi \vee \zeta$ or $\varphi = \psi \wedge \zeta$ is trivial.

- Case $\varphi = [\pi]_\diamond\psi$: Since $M', x \models [\pi]_\diamond\psi$, for every $\alpha, \beta \in \Pi_0^*$, $\sigma \in \Pi_0$, if $\alpha\sigma\beta \in \mathcal{L}(\pi)$ and $S'_\alpha(x, y)$ holds, then $R'_\sigma(y, z')$ holds for some z', and hence $R_\sigma(y, z)$ holds for some z. It follows that, for every $\alpha, \beta \in \Pi_0^*$ and $\sigma \in \Pi_0$, if $\alpha\sigma\beta \in \mathcal{L}(\pi)$ and $S_\alpha(x, y)$ holds, then $R_\sigma(y, z)$ holds for some z and $\{z \mid S'_\sigma(y, z)\} = \{z \mid S_\sigma(y, z)\}$. This together with $M', x \models [\pi]_\diamond\psi$ implies that $M, x \models [\pi]_\diamond\psi$. Since $M \leq_r M$ and $r(x, x')$ holds, it follows that $M, x' \models [\pi]_\diamond\psi$.

- Case $\varphi = \langle\pi\rangle\psi$: There exist $\sigma_1, \ldots, \sigma_k \in \Pi_0$ and $x_0 = x, x_1, \ldots, x_k \in W$ such that $\sigma_1 \ldots \sigma_k \in \mathcal{L}(\pi)$, $R'_{\sigma_i}(x_{i-1}, x_i)$ holds for $1 \leq i \leq k$, and $M', x_k \models \psi$. Let $x'_0 = x'$. For $1 \leq i \leq k$, choose x'_i as follows:

  - Case $R_{\sigma_i}(x_{i-1}, x_i)$ holds and $x_i = Next(x_{i-1}, \langle\sigma_i\rangle\zeta_i)$: We have that $\langle\sigma_i\rangle\zeta_i \in H(x_{i-1})$. By the proof of Lemma 7, $M, x_{i-1} \models \langle\sigma_i\rangle\zeta_i$. Since $M \leq_r M$ and $r(x_{i-1}, x'_{i-1})$ holds, it follows that $M, x'_{i-1} \models \langle\sigma_i\rangle\zeta_i$. Let x'_i be a state such that $R_{\sigma_i}(x'_{i-1}, x'_i)$ holds and $M, x'_i \models \zeta_i$. Thus, by (ii), $r(x_i, x'_i)$ holds.
  - Case $R_{\sigma_i}(x_{i-1}, x_i)$ does not hold and $x_i = Next_S(x_{i-1}, \sigma_i)$: There must exist ζ_i such that $Next(x_{i-1}, \langle\sigma_i\rangle\zeta_i)$ is defined. Choose x'_i as in the above subcase. Thus $R_{\sigma_i}(x'_{i-1}, x'_i)$ holds. By (iii), it follows that $r(x_i, x'_i)$ holds.

  Since $r(x_k, x'_k)$ holds and $M', x_k \models \psi$, by the inductive assumption, $M, x'_k \models \psi$. Since $R_{\sigma_i}(x'_{i-1}, x'_i)$ holds for every $1 \leq i \leq k$, it follows that $M, x'_0 \models \langle\pi\rangle\psi$, which means $M, x' \models \varphi$.

■

THEOREM 12. *Let P be a deterministic positive logic program in $PDL^{(0)}$, M a least pseudo-model of P, and φ a serial positive formula. Then $P \models \varphi$ iff $M \models \varphi$.*

Proof. Consider the "if" direction. Suppose that $M \models \varphi$. Let M' be an arbitrary Kripke model of P. As M' is also a pseudo-model, we have that $M \leq M'$. Hence $M' \models \varphi$. Therefore $P \models \varphi$.

Now consider the "only if" direction. Suppose that $P \models \varphi$.

We can assume that $M = \langle W, \tau, (R_\sigma)_{\sigma\in\Pi_0}, (S_\sigma)_{\sigma\in\Pi_0}, h\rangle$ is the pseudo-model constructed by Algorithm 5 for P. Let $M' = \langle W, \tau, (R'_\sigma)_{\sigma\in\Pi_0}, (S'_\sigma)_{\sigma\in\Pi_0}, h\rangle$ be the model corresponding to M. It is sufficient to show that $M' \models P$, because this implies that $M' \models \varphi$, and by Lemma 11, $M \models \varphi$.

Let H be the data structure of Algorithm 5 which specifies the contents of states of M. To show that $M' \models P$, we prove by induction on the construction of φ that if $\varphi \in H(u)$ then $M', u \models \varphi$. Suppose that $\varphi \in H(u)$. By the proof of Lemma 7, $M, u \models \varphi$. Using this, the only non-trivial case is when φ is of the form $B_1 \land \ldots \land B_k \to A$. Suppose that $M', u \models B_1 \land \ldots \land B_k$. By Lemma 11, $M, u \models B_1 \land \ldots \land B_k$. Since $M, u \models \varphi$, it follows that $M, u \models A$. This implies that $M', u \models A$ (because $R_\sigma \subseteq R'_\sigma$ and $S'_\sigma \subseteq S_\sigma$ for every $\sigma \in \Pi_0$). Therefore $M', u \models \varphi$. This completes the proof. ■

COROLLARY 13. *Let P be a deterministic positive logic program in PDL$^{(0)}$, M' the model corresponding to the pseudo-model constructed by Algorithm 5 for P, and φ a serial positive formula. Then $P \models \varphi$ iff $M' \models \varphi$.*

Proof. By the proof of the above theorem, $M' \models P$. Hence, $P \models \varphi$ implies $M' \models \varphi$. For the conversion, suppose that $M' \models \varphi$. Let M be the pseudo-model constructed by Algorithm 5 for P. By Lemma 11, $M \models \varphi$. Let M'' be an arbitrary model of P. Since M is a least pseudo-model of P, we have that $M \leq M''$, which implies $M'' \models \varphi$. Hence $P \models \varphi$. ■

5 Lower Bound

Proposition 6 states that, given a deterministic positive logic program P in PDL$^{(0)}$, a finite least pseudo-model of P can be constructed in exponential time and it has an exponential size (in the worst case). In this section, we give an example showing that, in general, this estimation is tight.

Let M and M' be pseudo-models. Define that $M \leq_\diamond M'$ if for every serial positive formula φ, $M \models \varphi$ implies $M' \models \varphi$. Define that $M \equiv_\diamond M'$ if $M \leq_\diamond M'$ and $M' \leq_\diamond M$.

LEMMA 14. *Let $M = \langle W, \tau, (R_\sigma)_{\sigma\in\Pi_0}, (S_\sigma)_{\sigma\in\Pi_0}, h\rangle$ be a finite pseudo-model such that, for every $\sigma \in \Pi_0$, R_σ is a function, i.e. $\forall x \exists! y R_\sigma(x, y)$. Let $M' = \langle W', \tau', (R'_\sigma)_{\sigma\in\Pi_0}, (S'_\sigma)_{\sigma\in\Pi_0}, h'\rangle$ be a finite pseudo-model such that $M' \equiv_\diamond M$. Then for every $w \in W$ reachable from τ via a path using $(R_\sigma)_{\sigma\in\Pi_0}$ there exists $w' \in W'$ such that $(M, w) \equiv_\diamond (M', w')$.*

Proof. We first show that, for every $\sigma \in \Pi_0$,

$$\forall x, x', y\ R_\sigma(x, y) \land ((M, x) \leq_\diamond (M', x')) \to \exists y'\ R'_\sigma(x', y') \land ((M, y) \leq_\diamond (M', y'))$$

Suppose that $R_\sigma(x,y)$ and $(M,x) \leq_\diamond (M',x')$. We show that there exists $y' \in W'$ such that $R'_\sigma(x',y')$ and $(M,y) \leq_\diamond (M',y')$. Suppose oppositely that for every $y' \in W'$ such that $R'_\sigma(x',y')$, $(M,y) \leq_\diamond (M',y')$ does not hold, i.e. there exists a serial positive formula $\varphi_{y'}$ such that $M,y \models \varphi_{y'}$ but $M',y' \not\models \varphi_{y'}$. Let φ be the conjunction of all such $\varphi_{y'}$. We have $M,x \models \langle\sigma\rangle\varphi$, while $M',x' \not\models \langle\sigma\rangle\varphi$, which contradicts the assumption that $(M,x) \leq_\diamond (M',x')$.

Similarly, we also have that, for every $\sigma \in \Pi_0$,

$$\forall x', y', x\ R'_\sigma(x',y') \wedge ((M',x') \leq_\diamond (M,x)) \rightarrow \\ \exists y\ R_\sigma(x,y) \wedge ((M',y') \leq_\diamond (M,y))$$

We now prove the claim of the lemma. It suffices to prove by induction on k that if $R_{\sigma_1\ldots\sigma_k}(\tau,w_k)$ holds then there exists $w'_k \in W'$ such that $(M,w_k) \equiv_\diamond (M',w'_k)$. The base case $k=0$ holds for $w'_0 = \tau'$. For the induction step, suppose that the hypothesis holds for k, and $R_{\sigma_{k+1}}(w_k,w_{k+1})$ holds for some $\sigma_{k+1} \in \Pi_0$. We show that there exists $w'_{k+1} \in W'$ such that $(M,w_{k+1}) \equiv_\diamond (M',w'_{k+1})$. Since $(M,w_k) \leq_\diamond (M',w'_k)$ and $R_{\sigma_{k+1}}(w_k,w_{k+1})$ holds, by (i), there exists $w'_{k+1} \in W'$ such that $R'_{\sigma_{k+1}}(w'_k,w'_{k+1})$ holds and $(M,w_{k+1}) \leq_\diamond (M',w'_{k+1})$. On the other hand, since $(M',w'_k) \leq_\diamond (M,w_k)$ and $R'_{\sigma_{k+1}}(w'_k,w'_{k+1})$ holds, by (ii), there exists $w''_{k+1} \in W$ such that $R_{\sigma_{k+1}}(w_k,w''_{k+1})$ holds and $(M',w'_{k+1}) \leq_\diamond (M,w''_{k+1})$. Since $R_{\sigma_{k+1}}$ is a function, $w''_{k+1} = w_{k+1}$, and hence $(M,w_{k+1}) \equiv_\diamond (M',w'_{k+1})$. ■

PROPOSITION 15. *Let* $\Pi_0 = \{a,b\}$ *and* $P = \{[(a\cup b)^*;a;(a\cup b)^{(n-1)}]p,$ $[(a\cup b)^*]\langle a\rangle\top,\ [(a\cup b)^*]\langle b\rangle\top\}$. *If* M' *is a pseudo-model characterizing* P *w.r.t. serial positive formulae (i.e. for every serial positive formula* φ, $P \models \varphi$ *iff* $M' \models \varphi$*), then* M' *has size* $2^{\Omega(n)}$.

Proof. Let $M = \langle W, \tau, (R_\sigma)_{\sigma\in\Pi_0}, (S_\sigma)_{\sigma\in\Pi_0}, h\rangle$ be the pseudo-model constructed by Algorithm 5 for P. It is easy to see that M satisfies the conditions stated in Lemma 14.

Let $\pi = (a\cup b)^*;a;(a\cup b)^{(n-1)}$. For $\alpha,\beta \in \Pi_0^*$, define that $\alpha \sim \beta$ if for every $\gamma \in \Pi_0^*$, $\alpha\gamma \in \mathcal{L}(\pi)$ iff $\beta\gamma \in \mathcal{L}(\pi)$. The equivalence relation $\sim$ has exactly 2^n abstract classes. Let $\alpha,\beta \in \Sigma^*$ and $\alpha \nsim \beta$. There exist w_α and w_β such that $R_\alpha(\tau,w_\alpha)$ and $R_\beta(\tau,w_\beta)$ hold. Since $\alpha \nsim \beta$, there exists $\gamma \in \Pi_0^*$ such that exactly one of $\alpha\gamma$ and $\beta\gamma$ belongs to $\mathcal{L}(\pi)$. Thus, $[\gamma]_\diamond p$ is true at exactly one of the worlds w_α and w_β. Hence $(M,w_\alpha) \not\equiv_\diamond (M,w_\beta)$. This implies that M contains at least 2^n states, which are reachable from τ (via paths using $(R_\sigma)_{\sigma\in\Pi_0}$) and not equivalent to each other. By Lemma 14, it follows that M' has at least 2^n states. ■

6 Further Work and Conclusions

Recall that if L is one of the basic propositional serial monomodal logics and P is a positive logic program in L then there exists a finite least L-model of P [7]. To obtain a similar result for PDL$^{(0)}$, we have restricted to the deterministic Horn fragment and used pseudo-models. The restriction is necessary to overcome the problem of nondeterminism caused by non-seriality, and pseudo-models are needed because that deterministic positive logic programs in PDL$^{(0)}$, e.g. $\{[\sigma]p\}$, do not always have least Kripke models. Pseudo-models satisfy the following expectations:

- A least pseudo-model M of a deterministic positive logic program P has the property that for every serial positive formula φ, $P \models \varphi$ iff $M \models \varphi$.
- Every deterministic positive logic program has a finite least pseudo-model.
- Given a pseudo-model M and a formula φ, the problem of checking $M \models \varphi$ is solvable in polynomial time in the sizes of M and φ.

The model M' corresponding to the least pseudo-model M constructed by Algorithm 5 for a deterministic positive logic program P also characterizes P w.r.t. serial positive formulae. However, M is more useful then M' in the aspect that, for every positive formula φ, $M \models \varphi$ implies $P \models \varphi$ (because M is less than or equal to every (pseudo-)model of P), while M' does not have such a property.

Our Algorithm 5 runs in exponential time and returns a pseudo-model with size $2^{O(n^2)}$. We have given a deterministic positive logic program such that every pseudo-model characterizing it w.r.t. serial positive formulae must have an exponential size. This does not imply that the (combined) complexity of checking satisfiability of deterministic Horn formulae is EXPTIME-complete. It is an open problem.

In the PDL$^{(0)}$-like description logic (PDL$^{(0)}$-Desc), programs of PDL$^{(0)}$ are used as role constructors. A TBox of that logic is a finite set of formulae of PDL$^{(0)}$, which are treated as global assumptions for all the states (but not as local assumptions of the current state τ). Apart from the TBox, a knowledge base in a description logic contains also an ABox, which is a set of facts of the form $p(a)$ or $R(a, b)$, where p is a "concept", a and b are "objects", and R is a "role name". In the terminology of PDL, p is an atomic proposition, a and b are states, and R can be assumed to be an atomic program. Note that objects in description logics correspond to states in PDL (and possible worlds in modal logics). The instance checking problem in PDL$^{(0)}$-Desc is stated as follows: given a TBox T and an ABox A of PDL$^{(0)}$-Desc, a concept C, and an object a, check whether a is an instance of C in every model of $T \cup A$ (i.e. whether $T \cup A \models C(a)$, where $\models$ reflects "global semantic consequence" in PDL$^{(0)}$-Desc). The data complexity of that problem is measured w.r.t. the size of A, while assuming that T, C, and a are fixed. Our claim is that if T is a deterministic positive logic program in PDL$^{(0)}$ then the data complexity of that problem is in PTIME. A formal proof of this will appear in an extension of this paper. The sketch is as follows:

- We construct a finite least pseudo-model M for the ABox A and the global assumptions T by starting with the graph corresponding to A

and proceeding in a similar way as Algorithm 5, except that T is added to the content of every state.

- Then $T \cup A \models C(a)$ iff $M, a \models C$.
- Since T is fixed, the size of M and the complexity of constructing M are bounded by a polynomial in the size of A.

Additionally, similarly to the extension [8] of [7], our method can be extended to develop declarative and procedural semantics for deterministic positive logic programs in first-order dynamic logic, which will be useful for logic programming about actions, time, belief, and knowledge. Goals to such programs are deterministic negative clauses. This is a line to combine modal and temporal logic programming and remains as a future work.

Acknowledgements

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

BIBLIOGRAPHY

[1] Ph. Balbiani, L. Fariñas del Cerro, and A. Herzig. Declarative semantics for modal logic programs. In *Proceedings of the 1988 International Conference on Fifth Generation Computer Systems*, pages 507–514. ICOT, 1988.
[2] M. Baldoni, L. Giordano, and A. Martelli. A framework for a modal logic programming. In *Joint International Conference and Symposium on Logic Programming*, pages 52–66. MIT Press, 1996.
[3] F. Debart, P. Enjalbert, and M. Lescot. Multimodal logic programming using equational and order-sorted logic. *Theoretical Comp. Science*, 105:141–166, 1992.
[4] R. Goré and L.A. Nguyen. A tableau system with automaton-labelled formulae for regular grammar logics. In B. Beckert, editor, *Proceedings of TABLEAUX 2005, LNAI 3702*, pages 138–152. Springer-Verlag, 2005.
[5] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000.
[6] U. Hustadt, B. Motik, and U. Sattler. Data complexity of reasoning in very expressive description logics. In L.P. Kaelbling and A. Saffiotti, editors, *IJCAI*, pages 466–471. Professional Book Center, 2005.
[7] L.A. Nguyen. Constructing the least models for positive modal logic programs. *Fundamenta Informaticae*, 42(1):29–60, 2000.
[8] L.A. Nguyen. A fixpoint semantics and an SLD-resolution calculus for modal logic programs. *Fundamenta Informaticae*, 55(1):63–100, 2003.
[9] L.A. Nguyen. Multimodal logic programming. To appear in TCS, 2006.
[10] A. Nonnengart. How to use modalities and sorts in Prolog. In C. MacNish, D. Pearce, and L.M. Pereira, editors, *Proceedings of JELIA'94, LNCS 838*, pages 365–378. Springer, 1994.

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