On the Complexity of Reasoning about Dynamic Policies

Stefan Göller*

Universität Stuttgart, FMI, Germany
goeller@informatik.uni-stuttgart.de

Abstract. We study the complexity of satisfiability for $\mathrm{DLP}^+_{\mathrm{dyn}}$, an expressive logic introduced by Demri that allows to reason about dynamic policies. $\mathrm{DLP}^+_{\mathrm{dyn}}$ extends the logic $\mathrm{DLP}_{\mathrm{dyn}}$ of Pucella and Weissman, which in turn extends van der Meyden's Dynamic Logic of Permission (DLP). $\mathrm{DLP}^+_{\mathrm{dyn}}$ generously enhances DLP and $\mathrm{DLP}_{\mathrm{dyn}}$ by allowing to update the policy set by adding or removing policy transitions, which are defined as a direct product of two sets, each specified by a formula of the logic itself. It is proven that satisfiability for $\mathrm{DLP}^+_{\mathrm{dyn}}$ is complete for deterministic exponential time. Our results close the complexity gap of satisfiability for $\mathrm{DLP}^+_{\mathrm{dyn}}$ from 2EXP, and for $\mathrm{DLP}_{\mathrm{dyn}}$ from NEXP, to EXP respectively, matching the EXP lower bound both inherit from Propositional Dynamic Logic (PDL). To prove the EXP upper bound for $\mathrm{DLP}^+_{\mathrm{dyn}}$, we first proceed by accurately identifying a suitable generalization of PDL, which allows to use compressed programs and then find a satisfiability preserving translation from $\mathrm{DLP}^+_{\mathrm{dyn}}$ to this extension of PDL. To finally show the EXP upper bound for $\mathrm{DLP}^+_{\mathrm{dyn}}$, we prove that satisfiability of our extension of PDL lies in EXP.

1 Introduction

Numerous applications contain a set of policies that, roughly speaking, describe what is prohibited and what is permitted. Policies arise in many contexts. For one, they can comprise control policies, thus specifying which agents are permitted to access resources. They can as well be legal policies, describing actions that are legally permitted. In the course of time, the policies of an application may change. This dynamic behaviour originates from the interaction between the application and its user(s). When changing policies, the system has to guarantee that unintentional side-effects do not occur. Furthermore, it is often not straightforward to decide whether to modify the current policy set or not, not to mention the duty of creating it. Various examples of typical practical scenarios are given in [10]. In order to allow comparison of different policies and reasoning about them, a variety of languages have been introduced, an overview is given in [15]. Van der Meyden's Dynamic Logic of Permission (DLP) is an example of an expressive logic that allows to reason about dynamic policies. Formally, it extends test-free Propositional Dynamic Logic (PDL) and allows to reason about a fixed policy set that describes the set of all permitted transitions of a system which is in turn modeled by a Kripke structure. In addition to test-free PDL, the logic DLP allows to ask queries of the kind 'does there exist a sequence of solely permitted transitions to some world

* The author is supported by the DFG project GELO.

where the property φ holds' and 'does every sequence of transitions to worlds satisfying φ solely consist of permitted transitions'. Extending DLP, the logic DLP_{dyn} of Pucella and Weissman [10] additionally allows to update the policy set by removing and adding transitions. These removed or added transitions are defined as the direct product of two sets of worlds, each specified by boolean combinations of atomic propositions. In [10], numerous applications are stated that demand for the possibility to update the policy set within the logic. The even more general logic $\text{DLP}^{+}_{\text{dyn}}$, introduced by Demri [3], allows to update the policy set by adding (via the `grant`-operator) and removing (via the `revoke`-operator) a direct product of world sets, but each specified by an *arbitrary* formula of the logic itself. In this paper, we focus on the computational complexity of an important algorithmic problem for $\text{DLP}^{+}_{\text{dyn}}$, namely satisfiability. For its fragment DLP_{dyn}, it is pointed out in [10] that the complexity of satisfiability is in NEXP. Focusing on naturalness, Demri gives a satisfiability preserving translation from $\text{DLP}^{+}_{\text{dyn}}$ to PDL [3]. By the presence of an exponential blowup in formula size in this translation, a 2EXP upper bound for satisfiability of $\text{DLP}^{+}_{\text{dyn}}$ was derived. However, if the depth of applied `grant` and `revoke` operators is bounded by some constant, an EXP upper bound was shown. Since the latter is the case for formulas of DLP, satisfiability for DLP was shown to be in EXP as a corollary. In this paper, we close the complexity gap for $\text{DLP}^{+}_{\text{dyn}}$ from 2EXP, and for DLP_{dyn} from NEXP, to EXP respectively, matching the EXP lower bound both inherit from PDL. An approach proposed in [3] to improve the complexity status of full $\text{DLP}^{+}_{\text{dyn}}$ is a polynomial time computable reduction to PDL enhanced with an operator $\oplus$ on programs (we call the resulting logic PDL$\oplus$ from now on) and then proving that satisfiability of PDL$\oplus$ lies in EXP. A program $\oplus(\pi, \varphi_1, \varphi_2)$, where π is a program and both φ_1 and φ_2 are formulas, relates pairs of worlds (x, y) of a Kripke structure, that are related via π such that additionally φ_1 holds in x or φ_2 holds in y. As we will remark later, PDL$\oplus$ is definable in PDL, but the size of the programs of the resulting PDL formula may grow exponentially in the size of the programs of the original PDL$\oplus$ formula. Alas, it turns out that a translation from $\text{DLP}^{+}_{\text{dyn}}$ to PDL$\oplus$ will not lead to an improvement of the complexity of $\text{DLP}^{+}_{\text{dyn}}$ – we prove that PDL$\oplus$ is 2EXP-complete. Thus, PDL$\oplus$ identifies a translatable fragment of both PDL with intersection [2] and of PDL, where programs are represented as dags, that is already hard for 2EXP. Yet to prove an EXP upper bound for $\text{DLP}^{+}_{\text{dyn}}$, we accurately identify a fragment PDL$\oplus[\mathbb{A}]$ of PDL$\oplus$, into which we translate $\text{DLP}^{+}_{\text{dyn}}$ and that we prove to lie in EXP. Our translation from $\text{DLP}^{+}_{\text{dyn}}$ to PDL$\oplus[\mathbb{A}]$ consists of a concise examination of how applied `grant` and `revoke` operators influence the truth of subformulas. For proving that PDL$\oplus[\mathbb{A}]$ lies in EXP, we translate an input formula of PDL$\oplus[\mathbb{A}]$ φ into an alternating Büchi tree automaton over infinite trees $\mathcal{A}(\varphi)$ and check the tree language of $\mathcal{A}(\varphi)$ for non-emptiness.

Firstly, our main contribution is to prove that $\text{DLP}^{+}_{\text{dyn}}$ and thus DLP_{dyn} is EXP-complete, which solves two open problems stated in [3]. The various applications of DLP and DLP_{dyn}, as listed in [12, 10], and the technical difficulties of reasoning about dynamic policies that arise, motivate an exact examination of the complexity of satisfiability of the more general $\text{DLP}^{+}_{\text{dyn}}$. Secondly, we believe that PDL with (restrictions on) the operator $\oplus$ is an interesting logic to study w.r.t. complexity, situated between EXP and 2EXP.

The paper is organized as follows. After introducing preliminaries in Section 2, we formally define $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ and related logics in Section 3. Known complexity results for comparable logics and the difficulty of handling $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ are summarized and discussed in Section 4. In Section 5, we give a satisfiability preserving translation from $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ to $\mathrm{PDL}{\oplus}[\mathbb{A}]$. An EXP upper bound for satisfiability of $\mathrm{PDL}{\oplus}[\mathbb{A}]$ is proven in Section 6. Finally, in Section 7, we show that satifiability for $\mathrm{PDL}{\oplus}$ is 2EXP-complete.

2 Preliminaries

If A and B are sets and $f : A \to B$ is a mapping, then for every subset $C \subseteq A$, we define $f(C) = \{f(c) \mid c \in C\}$. If $w = a_1 a_2 \cdots a_n$ is a string over the alphabet Σ and $a_i \in \Sigma$ for each $1 \leq i \leq n$, then $w^{(j)} = a_1 \cdots a_j$ denotes the *j-th prefix* of w for every $0 \leq j \leq n$, where $w^{(0)} = \varepsilon$. For a string w, let $|w|$ denote the *length* of w. If X is a set, $R \subseteq X \times X$ is a binary relation over X and $A, B \subseteq X$, then $\oplus(R, A, B) = R \cap ((A \times X) \cup (X \times B))$. If X and Y are sets with $X \cap Y = \emptyset$, then $X \uplus Y$ denotes the union of X and Y and recalls the fact that X and Y are disjoint. For every $l, k \in \mathbb{N}$, define $[k] = \{1, \ldots, k\}$, and $[l, k] = \{l, l+1, \ldots, k\}$. Let us introduce NFAs. An *NFA* is a tuple $A = (Q, \Sigma, q_0, \delta, F)$, where (i) Q is a finite set of *states*, (ii) Σ is a finite *alphabet*, (iii) $q_0 \in Q$ is an *initial state*, (iv) $F \subseteq Q$ is a set of *final states*, and (v) $\delta : Q \times \Sigma \to 2^Q$ is a *transition function*. We abbreviate $q' \in \delta(q, a)$ by $q \overset{a}{\Rightarrow}_A q'$. Next, we extend $\Rightarrow_A$ to words over Σ. For all $q \in Q$ we have $q \overset{\varepsilon}{\Rightarrow}_A q$. If $w \in \Sigma^*$, $a \in \Sigma$, $q \overset{w}{\Rightarrow}_A q'$, and $q' \overset{a}{\Rightarrow}_A q''$, then $q \overset{wa}{\Rightarrow}_A q''$. Let $L(A) = \{w \in \Sigma^* \mid q_0 \overset{w}{\Rightarrow}_A q \text{ for some } q \in F\}$ denote the *language* of A.

3 Logic

For the rest of the paper, fix some countable set of *atomic propositions* $\mathbb{P}$ and some countable set of *atomic programs* $\mathbb{A}$.

3.1 $\mathbf{DLP}^{+}_{\mathbf{dyn}}$ and its fragments $\mathbf{DLP}_{\mathbf{dyn}}$ and DLP

Formulas φ and *programs* π of the logic $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ are given by the following grammar, where p ranges over $\mathbb{P}$ and a ranges over $\mathbb{A}$:

$$\begin{array}{lcl} \varphi & ::= & p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \langle\pi\rangle\,\varphi \mid \mathtt{perm}(\pi)\varphi \mid \mathtt{fperm}(\pi)\varphi \mid \\ & & \mathtt{grant}(\varphi_1, \varphi_2)\varphi_3 \mid \mathtt{revoke}(\varphi_1, \varphi_2)\varphi_3 \\ \pi & ::= & a \mid \pi_1 \cup \pi_2 \mid \pi_1 \circ \pi_2 \mid \pi^* \mid \varphi? \end{array}$$

We introduce the following abbreviations: $\varphi_1 \wedge \varphi_2 = \neg(\neg\varphi_1 \vee \neg\varphi_2)$, $\mathtt{true} = p \vee \neg p$ for some $p \in \mathbb{P}$, $\mathtt{false} = \neg\mathtt{true}$, and $[\pi]\varphi = \neg\langle\pi\rangle\neg\varphi$. Let Φ denote the set of all $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ formulas and let $\mathrm{Test} = \{\varphi? \mid \varphi \in \Phi\}$ denote the set of all $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ *test programs*. The logic $\mathrm{DLP}_{\mathrm{dyn}}$ is the syntactic fragment of $\mathrm{DLP}^{+}_{\mathrm{dyn}}$, where the first two components of `grant` and `revoke` formulas are restricted to be boolean combinations of atomic propositions and where test programs do not occur. The logic DLP is the fragment of $\mathrm{DLP}_{\mathrm{dyn}}$, where the operators `grant` and `revoke` do not occur. For every $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ program π let $L(\pi)$ denote the *regular language* of π interpreted as a regular expression over some finite subset from $\mathbb{A} \cup \mathrm{Test}$. A *Kripke structure* is a tuple $(X, \{\rightarrow_a \mid a \in \mathbb{A}\}, \varrho)$, where X is a set of *worlds*, $\rightarrow_a \subseteq X \times X$ is a binary relation for each $a \in \mathbb{A}$, and $\varrho : X \rightarrow 2^{\mathbb{P}}$ assigns to each world $x \in X$ a set of atomic propositions $\varrho(x)$. An *extended Kripke* structure is a tuple $(X, \{\rightarrow_a \mid a \in \mathbb{A}\}, \varrho, P)$, where $(X, \{\rightarrow_a \mid a \in \mathbb{A}\}, \varrho)$ is a Kripke structure and $P \subseteq X \times X$ is a binary relation that we call *policy set*. If $\mathrm{op} \in \{\cup, \setminus\}$, $\mathcal{K} = (X, \{\rightarrow_a \mid a \in \mathbb{A}\}, \varrho, P)$ is an extended Kripke structure, and $A, B \subseteq X$, then $\mathcal{K} \upharpoonright (A, B, \mathrm{op}) = (X, \{\rightarrow_a \mid a \in \mathbb{A}\}, \varrho, P \;\mathrm{op}\; (A \times B))$. Fix some extended Kripke structure $\mathcal{K} = (X, \{\rightarrow_a \mid a \in \mathbb{A}\}, \varrho, P)$. Then, for each atomic/test program π, we define a binary relation $[\![\pi]\!]_{\mathcal{K}} \subseteq X \times X$ and for each formula $\varphi \in \Phi$ we define a subset $[\![\varphi]\!]_{\mathcal{K}} \subseteq X$ inductively as follows, where $p \in \mathbb{P}$ and $a \in \mathbb{A}$:

$$
\begin{aligned}
[\![a]\!]_{\mathcal{K}} &= \rightarrow_a \\
[\![\varphi?]\!]_{\mathcal{K}} &= \{(x,x) \mid x \in [\![\varphi]\!]_{\mathcal{K}}\} \\
[\![p]\!]_{\mathcal{K}} &= \{x \in X \mid p \in \varrho(x)\} \\
[\![\neg\varphi]\!]_{\mathcal{K}} &= X \setminus [\![\varphi]\!]_{\mathcal{K}} \\
[\![\varphi_1 \vee \varphi_2]\!]_{\mathcal{K}} &= [\![\varphi_1]\!]_{\mathcal{K}} \cup [\![\varphi_2]\!]_{\mathcal{K}}
\end{aligned}
$$

$$
\begin{aligned}
[\![\langle\pi\rangle\varphi]\!]_{\mathcal{K}} = \{x \in X \mid\ & \text{there exist } x_0, x_1, \ldots, x_n \in X \text{ with} \\
& x = x_0, x_n \in [\![\varphi]\!]_{\mathcal{K}}, \text{ and there exist} \\
& A_1 \cdots A_n \in L(\pi) \text{ s.t. for all } 1 \leq i \leq n \\
& \text{we have } (x_{i-1}, x_i) \in [\![A_i]\!]_{\mathcal{K}}\}
\end{aligned}
$$

$$
\begin{aligned}
[\![\mathtt{perm}(\pi)\varphi]\!]_{\mathcal{K}} = \{x \in X \mid\ & \text{there exist } x_0, x_1, \ldots, x_n \in X \text{ with} \\
& x = x_0, x_n \in [\![\varphi]\!]_{\mathcal{K}}, \text{ and there exist} \\
& A_1 \cdots A_n \in L(\pi) \text{ s.t. for all } 1 \leq i \leq n \\
& \text{we have } (x_{i-1}, x_i) \in [\![A_i]\!]_{\mathcal{K}} \text{ and} \\
& A_i \in \mathbb{A} \text{ implies } (x_{i-1}, x_i) \in P\}
\end{aligned}
$$

$$
\begin{aligned}
[\![\mathtt{fperm}(\pi)\varphi]\!]_{\mathcal{K}} = \{x \in X \mid\ & \text{there does not exist} \\
& x_0, x_1, \ldots, x_n \in X \text{ and } A_1 \cdots A_n \in L(\pi) \text{ with} \\
& x = x_0, x_n \in [\![\varphi]\!]_{\mathcal{K}} \text{ and } (x_{i-1}, x_i) \in [\![A_i]\!]_{\mathcal{K}} \\
& \text{for all } 1 \leq i \leq n \text{ such that } A_j \in \mathbb{A} \text{ and} \\
& (x_{j-1}, x_j) \notin P \text{ for some } 1 \leq j \leq n\}
\end{aligned}
$$

$$
\begin{aligned}
[\![\mathtt{grant}(\varphi_1,\varphi_2)\varphi_3]\!]_{\mathcal{K}} &= [\![\varphi_3]\!]_{\mathcal{K}\upharpoonright([\![\varphi_1]\!]_{\mathcal{K}}, [\![\varphi_2]\!]_{\mathcal{K}}, \cup)} \\
[\![\mathtt{revoke}(\varphi_1,\varphi_2)\varphi_3]\!]_{\mathcal{K}} &= [\![\varphi_3]\!]_{\mathcal{K}\upharpoonright([\![\varphi_1]\!]_{\mathcal{K}}, [\![\varphi_2]\!]_{\mathcal{K}}, \setminus)}
\end{aligned}
$$

We abbreviate $x \in [\![\varphi]\!]_{\mathcal{K}}$ by $(\mathcal{K}, x) \models \varphi$. If for some state $x \in X$ we have $(\mathcal{K}, x) \models \varphi$ then $\mathcal{K}$ is a *model* for φ. We say that a formula φ is *satisfiable* if there exists some model for φ. The size $|\varphi|$ of a $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ formula φ and the size $|\pi|$ of a $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ program π is inductively defined as follows: $|p| = |a| = 1$ for every $p \in \mathbb{P}$ and for every $a \in \mathbb{A}$, $|\varphi_1 \vee \varphi_2| = |\varphi_1| + |\varphi_2| + 1$, $|\neg\varphi| = |\varphi| + 1$, $|\langle\pi\rangle\varphi| = |\pi| + |\varphi| + 1$, $|\texttt{perm}(\pi)\varphi| = |\texttt{fperm}(\pi)\varphi| = |\pi| + |\varphi| + 1$, $|\texttt{grant}(\varphi_1, \varphi_2)\varphi_3| = |\texttt{revoke}(\varphi_1, \varphi_2)\varphi_3| = |\varphi_1| + |\varphi_2| + |\varphi_3| + 1$, $|\pi_1 \cup \pi_2| = |\pi_1 \circ \pi_2| = |\pi_1| + |\pi_2| + 1$, $|\pi^*| = |\pi| + 1$, and $|\varphi?| = |\varphi| + 1$. The set $\mathrm{subf}(\varphi)$ of *subformulas* of a formula φ and the set of *subformulas* $\mathrm{subf}(\pi)$ of a program π is inductively defined as follows: (i) $\mathrm{subf}(p) = \{p\}$ for all $p \in \mathbb{P}$, (ii) $\mathrm{subf}(\neg\varphi) = \{\neg\varphi\} \cup \mathrm{subf}(\varphi)$, (iii) $\mathrm{subf}(\varphi_1 \vee \varphi_2) = \{\varphi_1 \vee \varphi_2\} \cup \mathrm{subf}(\varphi_1) \cup \mathrm{subf}(\varphi_2)$, (iv) if $\varphi = \langle\pi\rangle\psi$, $\varphi = \texttt{perm}(\pi)\psi$, or $\varphi = \texttt{fperm}(\pi)\psi$, then $\mathrm{subf}(\varphi) = \{\varphi\} \cup \mathrm{subf}(\pi) \cup \mathrm{subf}(\psi)$, (v) if $\varphi = \texttt{grant}(\varphi_1, \varphi_2)\varphi_3$ or $\varphi = \texttt{revoke}(\varphi_1, \varphi_2)\varphi_3$, then $\mathrm{subf}(\varphi) = \{\varphi\} \cup \bigcup_{i=1}^{3} \mathrm{subf}(\varphi_i)$, (vi) $\mathrm{subf}(a) = \emptyset$ for all $a \in \mathbb{A}$, (vii) $\mathrm{subf}(\pi_1 \cup \pi_2) = \mathrm{subf}(\pi_1 \circ \pi_2) = \mathrm{subf}(\pi_1) \cup \mathrm{subf}(\pi_2)$, $\mathrm{subf}(\pi^*) = \mathrm{subf}(\pi)$, and finally (viii) $\mathrm{subf}(\varphi?) = \mathrm{subf}(\varphi)$.

3.2 PDL and its compressed variants PDL$\oplus$ and PDL$\oplus[\mathbb{A}]$

Formulas φ and *programs* π of the logic PDL$\oplus$ are given by the following grammar, where p ranges over $\mathbb{P}$ and a ranges over $\mathbb{A}$:

$$
\begin{array}{lcl}
\varphi & ::= & p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \langle\pi\rangle\,\varphi \\
\pi & ::= & a \mid \pi_1 \cup \pi_2 \mid \pi_1 \circ \pi_2 \mid \pi^* \mid \varphi? \mid \oplus(\pi, \varphi_1, \varphi_2)
\end{array}
$$

Formulas and programs of PDL$\oplus$ are interpreted in Kripke structures (hence policy sets do not occur). If $\mathcal{K} = (X, \{\rightarrow_a \mid a \in \mathbb{A}\}, \varrho)$ is a Kripke structure, then for each program π, we can assign a binary relation $[\![\pi]\!]_{\mathcal{K}} \subseteq X \times X$, which is defined homomorphic for $\cup, \circ$, and $*$, and where the semantics of the program operator $\oplus$ is defined as $[\![\oplus(\pi, \varphi_1, \varphi_2)]\!]_{\mathcal{K}} = \oplus([\![\pi]\!]_{\mathcal{K}}, [\![\varphi_1]\!]_{\mathcal{K}}, [\![\varphi_2]\!]_{\mathcal{K}})$. The size of $\oplus(\pi, \varphi_1, \varphi_2)$ is defined as $|\oplus(\pi, \varphi_1, \varphi_2)| = 1 + |\pi| + |\varphi_1| + |\varphi_2|$. If $\pi = \oplus(\pi', \varphi_1, \varphi_2)$, then the set of subformulas $\mathrm{subf}(\pi)$ is defined as $\mathrm{subf}(\pi) = \mathrm{subf}(\pi') \cup \mathrm{subf}(\varphi_1) \cup \mathrm{subf}(\varphi_2)$. Similarly as above, a Kripke structure $(X, \{\rightarrow_a \mid a \in \mathbb{A}\}, \varrho)$ is a *model* for a PDL$\oplus$ formula φ, if for some state $x \in X$ we have $(\mathcal{K}, x) \models \varphi$. A PDL$\oplus$ formula φ is *satisfiable*, if there exists some model for φ. The logic PDL$\oplus[\mathbb{A}]$ is the syntactic fragment of PDL$\oplus$, where the program arguments of a $\oplus$ program must either be an atomic program or a $\oplus$ program itself. More formally, *formulas* φ, *basic programs* α and *programs* π of PDL$\oplus[\mathbb{A}]$ are given by the following grammar, where p ranges over $\mathbb{P}$ and a ranges over $\mathbb{A}$:

$$
\begin{array}{lcl}
\varphi & ::= & p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \langle\pi\rangle\,\varphi \\
\alpha & ::= & a \mid \oplus(\alpha, \varphi_1, \varphi_2) \\
\pi & ::= & \alpha \mid \pi_1 \cup \pi_2 \mid \pi_1 \circ \pi_2 \mid \pi^* \mid \varphi?
\end{array}
$$

The logic PDL is the syntactic fragment of PDL$\oplus$ without the $\oplus$ operator.

Remark 1. A PDL$\oplus$ program $\oplus(\pi, \varphi_1, \varphi_2)$ can be translated into an equivalent PDL program $\|\oplus(\pi, \varphi_1, \varphi_2)\|$ as follows, where $\|\pi\|$, $\|\varphi_1\|$, and $\|\varphi_2\|$ are inductively the translations for π, φ_1, and φ_2 respectively:

$$\|\oplus(\pi, \varphi_1, \varphi_2)\| \quad = \quad (\|\varphi_1\|? \circ \|\pi\| \cup \|\pi\| \circ \|\varphi_2\|?)$$

Hence, the logics PDL$\oplus$, PDL$\oplus[\mathbb{A}]$, and PDL are equi-expressive. However, by the presence of $\oplus$ programs, the above translation might cause an exponential blowup. Thus, the three logics may differ in succintness. In fact, our 2EXP-completeness result for PDL$\oplus$ shown in Section 7 implies that there does not exist a satisfiability preserving translation from PDL$\oplus$ to PDL that is computable in polynomial time.

4 Known results and difficulties of reasoning about $\mathbf{DLP}^{+}_{\mathbf{dyn}}$ and related logics

In this section, we state known results and explain some difficulties of determining the complexity of reasoning about $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ and logics related to it.

The *satisfiability problem* asks, given a formula φ of any of the logics introduced above, whether φ is satisfiable. By Fischer/Ladner and Pratt it is known:

Theorem 1 ([4, 9]). *Satisfiability for PDL is* EXP*-complete.*

Focusing on a natural translation from $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ to PDL, Demri has recently shown:

Theorem 2 ([3]). *There exists a satisfiability preserving reduction from DLP$^{+}_{\mathrm{dyn}}$ to PDL computable in polynomial time.*

Since the size of the resulting PDL formula in the proof of Theorem 2 is exponential in the size of the original $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ formula, the following theorem holds.

Theorem 3 ([3]). *Satisfiability for DLP$^{+}_{\mathrm{dyn}}$ is in* 2EXP.

For its fragment $\mathrm{DLP}_{\mathrm{dyn}}$, the following upper bound is known.

Theorem 4 ([10]). *Satisfiability for DLP$_{\mathrm{dyn}}$ is in* NEXP.

Let us summarize the difficulties of identifying the exact complexity of $\mathrm{DLP}^{+}_{\mathrm{dyn}}$. By the presence of the operators `grant` and `revoke`, the truth of a formula may depend on the truth of a subformula in some modified extended Kripke structure. This behaviour is very much in the flavor of sabotage modal logic of van Benthem [11]. The latter is basically modal logic enhanced with a sabotage operator $\langle - \rangle$. A formula $\langle - \rangle\varphi$ holds in a world x of a Kripke structure $\mathcal{K}$ with state set X, if φ holds in x in $\mathcal{K}'$, where $\mathcal{K}'$ is a Kripke structure that emerges from $\mathcal{K}$ by removing some world from $X \setminus \{x\}$. The decidability status for sabotage modal logic is unknown so far. Yet, a variant of it, in which the sabotage operator requires to remove some labeled transition instead of some world, has been proven undecidable by Löding and Rohde [7]. At first glance, one could think that the situation for $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ is even worse, since transitions can be removed *and* added and since moreover these transitions can be specified in the logic itself. Interestingly, it turns out, that precisely the fact that the updated transitions are specified in the logic itself, allows translations to other logics that are more manageable w.r.t. satisfiability. So, one promising approch to decide $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ in EXP was a translation into PDL with intersection and negation of atomic programs given in [3]. This translation has the property that the width of nested intersection of the resulting formulas is bounded by some constant. Firstly, a precise analysis of [5] yields that satisfiability of PDL with

the intersection on programs, where formulas have bounded intersection width, is in EXP. Secondly, a result from [8] states that PDL with negation of atomic programs is in EXP too. But unfortunately, PDL with intersection *and* negation of atomic programs has proven to be undecidable recently [5], even when restricting to formulas of constant intersection width. A proposal of Demri [3] to improve the 2EXP upper bound for $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ is to give a polynomial translation from $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ to PDL$\oplus$ and to try to decide satisfiability of PDL$\oplus$ in EXP. However, we will prove in Section 7, that PDL$\oplus$ is complete for 2EXP.
Wrapping up, all mentioned translations have the drawback that either the target logic was too hard w.r.t. complexity or the translations have a blowup in formula size. Nevertheless, our solution to decide $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ in deterministic exponential time is to find a tricky translation into PDL$\oplus$'s adequate fragment PDL$\oplus[\mathbb{A}]$ and to show that PDL$\oplus[\mathbb{A}]$ is in EXP. By combining the EXP-hardness $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ inherits from PDL [4], we state the main result of this paper.

Theorem 5. *Satisfiability for $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ is* EXP*-complete.*

As stated above, for proving Theorem 5, we first give a satisfiability preserving translation from $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ to PDL$\oplus[\mathbb{A}]$, that can be computed in polynomial time in Section 5. An EXP upper bound for PDL$\oplus[\mathbb{A}]$ is proven in Section 6.

5 A translation from $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ to PDL$\oplus[\mathbb{A}]$

In this section, we give a satisfiability preserving translation from $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ to PDL$\oplus[\mathbb{A}]$ that is computable in polynomial time. In this translation, a precise analysis of how applied `grant` and `revoke` operators influence the truth of subformulas, allows to handle $\mathrm{DLP}^{+}_{\mathrm{dyn}}$. In parts, it combines some ideas from [3] and [6].
First, we introduce a notion of certain modified Kripke structures. Recall that Φ denotes the set of all $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ formulas. Let $\Sigma = (\Phi \times \Phi \times \{\cup, \setminus\})^*$. For each $\sigma = (\theta_1, \theta'_1, \mathrm{op}_1) \cdots (\theta_k, \theta'_k, \mathrm{op}_k) \in \Sigma$, where $k \geq 0$, let $U_\sigma = \{m \in [k] \mid \mathrm{op}_m = \cup\}$ and $M_\sigma = \{m \in [k] \mid \mathrm{op}_m = \setminus\}$. If $\mathcal{K}$ is an extended Kripke structure, then for every $\sigma \in \Sigma$ define the extended Kripke structure $\mathcal{K} \upharpoonright \sigma$, by the length of σ, inductively as follows: $\mathcal{K} \upharpoonright \varepsilon = \mathcal{K}$ and $\mathcal{K} \upharpoonright \sigma(\psi_1, \psi_2, \mathrm{op}) = (\mathcal{K} \upharpoonright \sigma) \upharpoonright ([\![\psi_1]\!]_{\mathcal{K}\upharpoonright\sigma}, [\![\psi_2]\!]_{\mathcal{K}\upharpoonright\sigma}, \mathrm{op})$ for all $\psi_1, \psi_2 \in \Phi$ and $\mathrm{op} \in \{\cup, \setminus\}$.

Remark 2. If $\mathcal{K}$ is an extended Kripke structure and $\sigma \in \Sigma$, then the extended Kripke structures $\mathcal{K}$ and $\mathcal{K} \upharpoonright \sigma$ can only differ in their policy set. Thus, for all $a \in \mathbb{A}$ we have $[\![a]\!]_{\mathcal{K}} = [\![a]\!]_{\mathcal{K}\upharpoonright\sigma}$ and for all $p \in \mathbb{P}$ we have $[\![p]\!]_{\mathcal{K}} = [\![p]\!]_{\mathcal{K}\upharpoonright\sigma}$.

For a formula $\varphi \in \Phi$, let $\mathrm{Occ}(\varphi)$ denote the set of all occurrences of subformulas of φ and for each $\psi \in \mathrm{Occ}(\varphi)$, define the unique sequence $\sigma(\psi) \in \Sigma$ that we get by considering the `grant` and `revoke` operators that occur along the path from φ to ψ in the syntax tree of φ. We define, in a top down manner, $\sigma : \mathrm{Occ}(\varphi) \to \Sigma$ as follows:

- $\sigma(\varphi) = \varepsilon$
- If $\psi = \neg\chi$, then $\sigma(\chi) = \sigma(\psi)$.
- If $\psi = \psi_1 \vee \psi_2$, then $\sigma(\psi_1) = \sigma(\psi_2) = \sigma(\psi)$.

– If $\psi = \langle\pi\rangle\chi$, $\psi = \mathtt{perm}(\pi)\chi$, or $\psi = \mathtt{fperm}(\pi)\chi$, then $\sigma(\chi) = \sigma(\psi') = \sigma(\psi)$ for every test program $\psi'?$ of π.
– If $\psi = \mathtt{grant}(\psi_1, \psi_2)\chi$, then $\sigma(\psi_1) = \sigma(\psi_2) = \sigma(\psi)$, and $\sigma(\chi) = \sigma(\psi)(\psi_1, \psi_2, \cup)$.
– If $\psi = \mathtt{revoke}(\psi_1, \psi_2)\chi$, then $\sigma(\psi_1) = \sigma(\psi_2) = \sigma(\psi)$, and $\sigma(\chi) = \sigma(\psi)(\psi_1, \psi_2, \setminus)$.

Before giving the translation from $\mathrm{DLP}^+_{\mathrm{dyn}}$ to PDL$\oplus[\mathbb{A}]$, the following lemma characterizes the policy set of $\mathcal{K} \restriction \sigma$, for every $\sigma \in \Sigma$. Its proof is by induction on k.

Lemma 1. *Let $\mathcal{K} = (X, \{\rightarrow_a \mid a \in \mathbb{A}\}, \varrho, P)$ be an extended Kripke structure, $\sigma = (\theta_1, \theta'_1, \mathrm{op}_1) \cdots (\theta_k, \theta'_k, \mathrm{op}_k) \in \Sigma$ (with $k \geq 0$), $\mathcal{K} \restriction \sigma = (X, \{\rightarrow_a \mid a \in \mathbb{A}\}, \varrho, P_\sigma)$ and $a \in \mathbb{A}$.*
Then, for all $(x, y) \in X \times X$, we have $(x, y) \in [\![a]\!]_{\mathcal{K}\restriction\sigma} \cap P_\sigma$ if and only if there exists some $u \in \{0\} \uplus U_\sigma$ such that for all $m \in M_\sigma \cap [u, k]$ we have $(x, y) \in [\![\oplus(a, \neg\theta_m, \neg\theta'_m)]\!]_{\mathcal{K}\restriction\sigma^{(m-1)}}$ and either (i) $u = 0$ and $(x, y) \in P \cap [\![a]\!]_{\mathcal{K}}$, or (ii) $u \in U_\sigma$ and $(x, y) \in [\![a]\!]_{\mathcal{K}} \cap ([\![\theta_u]\!]_{\mathcal{K}\restriction\sigma^{(u-1)}} \times [\![\theta'_u]\!]_{\mathcal{K}\restriction\sigma^{(u-1)}})$.
Conversely, for all $(x, y) \in X \times X$, we have $(x, y) \in [\![a]\!]_{\mathcal{K}\restriction\sigma} \setminus P_\sigma$ if and only if there exists some $m \in \{0\} \uplus M_\sigma$ such that for all $u \in U_\sigma \cap [m, k]$ we have $(x, y) \in [\![\oplus(a, \neg\theta_u, \neg\theta'_u)]\!]_{\mathcal{K}\restriction\sigma^{(u-1)}}$ and either (i) $m = 0$ and $(x, y) \in [\![a]\!]_{\mathcal{K}} \setminus P$, or (ii) $m \in M_\sigma$ and $(x, y) \in [\![a]\!]_{\mathcal{K}} \cap ([\![\theta_m]\!]_{\mathcal{K}\restriction\sigma^{(m-1)}} \times [\![\theta'_m]\!]_{\mathcal{K}\restriction\sigma^{(m-1)}})$.

Let us turn to our translation. For this, fix some $\mathrm{DLP}^+_{\mathrm{dyn}}$ formula φ over atomic programs A and over atomic propositions P for the rest of this section. Let $\{\psi_1, \ldots, \psi_n\}$ be an enumeration of $\mathrm{Occ}(\varphi)$ and assume $\psi_n = \varphi$. Let $\mathsf{A}' = \mathsf{A} \uplus \overline{\mathsf{A}}$ and $\mathsf{P}' = \mathsf{P} \uplus \{p_1, \ldots, p_n\}$. Below, we also write $p(\psi_i)$ for p_i whenever $\psi_i \in \mathrm{Occ}(\varphi)$. If $\sigma = (\theta_1, \theta'_1, \mathrm{op}_1) \cdots (\theta_k, \theta'_k, \mathrm{op}_k) \in \sigma(\mathrm{Occ}(\varphi))$, then for every subset $S = \{j_1, \ldots, j_l\} \subseteq [k]$, where $j_1 < j_2 < \cdots < j_l$, and every PDL$\oplus[\mathbb{A}]$ program ζ, define the PDL$\oplus$ program $\zeta^S = \zeta^S_l$, where $\zeta^S_0 = \zeta$ and $\zeta^S_h = \oplus(\zeta^S_{h-1}, \neg p(\theta_{j_h}), \neg p(\theta'_{j_h}))$ for all $1 \leq h \leq l$. We will build a PDL$\oplus[\mathbb{A}]$ formula φ' over the atomic programs A' and over the atomic propositions P' such that φ is satisfiable if and only if φ' is satisfiable. Intuitively, if $\mathcal{K}$ is a model for φ with policy set P and $\mathcal{K}'$ is a model of φ', think of the relation $[\![a]\!]_{\mathcal{K}'}$ as $[\![a]\!]_{\mathcal{K}} \cap P$ and of $[\![\overline{a}]\!]_{\mathcal{K}'}$ as $[\![a]\!]_{\mathcal{K}} \setminus P$. Let us first, for every $\psi_i \in \mathrm{Occ}(\varphi)$, define the PDL$\oplus[\mathbb{A}]$ formula $\|\psi_i\|$ over the atomic propositions P' and over the atomic programs A' inductively as follows:

– If $\psi_i = p$ for some $p \in \mathsf{P}$, then $\|\psi_i\| = p$.
– If $\psi_i = \neg\psi_j$, then $\|\psi_i\| = \neg p_j$.
– If $\psi_i = \psi_j \vee \psi_k$, then $\|\psi_i\| = p_j \vee p_k$.
– If $\psi_i = \mathtt{grant}(\psi_j, \psi_k)\psi_l$ or $\psi_i = \mathtt{revoke}(\psi_j, \psi_k)\psi_l$, then $\|\psi_i\| = p_l$.
– If $\psi_i = \langle\pi\rangle\psi_j$, then $\|\psi_i\| = \langle T(\pi)\rangle p_j$, where $T(\pi)$ is homomorphic on $\cup, \circ$, and on $*$, $T(\psi_k?) = p_k?$ for every test program $\psi_k?$, and $T(a) = a \cup \overline{a}$ for every $a \in \mathsf{A}$.
– Assume $\psi_i = \mathtt{perm}(\pi)\psi_j$ and $\sigma = \sigma(\psi_i) = (\theta_1, \theta'_1, \mathrm{op}_1) \cdots (\theta_k, \theta'_k, \mathrm{op}_k)$. Recall that $U_\sigma = \{u \in [k] \mid \mathrm{op}_u = \cup\}$ and $M_\sigma = \{m \in [k] \mid \mathrm{op}_m = \setminus\}$. Then, we define $\|\psi_i\| = \langle T^\forall(\pi)\rangle p_j$, where $T^\forall(\pi)$ is homomorphic on $\cup, \circ$, and on $*$, $T^\forall(\psi_k?) = p_k?$ for every test program $\psi_k?$. For every $a \in \mathsf{A}$, we define [1]:

$$T^\forall(a) \quad = \quad a^{M_\sigma} \cup \bigcup_{u \in U_\sigma} p(\theta_u)? \circ (a \cup \overline{a})^{M_\sigma \cap [u,k]} \circ p(\theta'_u)?$$

[1] To avoid lengthy notations, the programs $T^\forall(a)$ and $T^\exists(a)$ for $a \in \mathsf{A}$ are not PDL$\oplus[\mathbb{A}]$ programs (since the $\oplus$ operator is applied to a non-atomic programs of the kind $a \cup \overline{a}$, where

Via application of Lemma 1, the intention behind $T^{\forall}(a)$ is to describe the relation $[\![a]\!]_{\mathcal{K}\restriction\sigma} \cap P_\sigma$, where P_σ is the policy set of $\mathcal{K} \restriction \sigma$.

– Assume $\psi_i = \texttt{fperm}(\pi)\psi_j$ and $\sigma = \sigma(\psi_i) = (\theta_1, \theta'_1, \mathrm{op}_1) \cdots (\theta_k, \theta'_k, \mathrm{op}_k)$. Then, we define $\|\psi_i\| = \neg\langle T^{\exists}(\pi)\rangle p_j$, where $T^{\exists}(\pi)$ is inductively defined as follows:
  - $T^{\exists}(\pi_1 \cup \pi_2) = T^{\exists}(\pi_1) \cup T^{\exists}(\pi_2)$
  - $T^{\exists}(\pi_1 \circ \pi_2) = T^{\exists}(\pi_1) \circ T(\pi_2) \;\cup\; T(\pi_1) \circ T^{\exists}(\pi_2)$, where T is defined as above.
  - $T^{\exists}(\pi^*) = T(\pi^*) \circ T^{\exists}(\pi) \circ T(\pi^*)$, where T is defined as above.
  - $T^{\exists}(\psi_k?) = \texttt{false}?$
  - For every $a \in \mathsf{A}$, we define[1]:

$$T^{\exists}(a) \qquad = \qquad \overline{a}^{U_\sigma} \;\cup\; \bigcup_{m \in M_\sigma} p(\theta_m)? \circ (a \cup \overline{a})^{U_\sigma \cap [m,k]} \circ p(\theta'_m)?$$

Via application of Lemma 1, the intention behind $T^{\exists}(a)$ is to describe the relation $[\![a]\!]_{\mathcal{K}\restriction\sigma} \setminus P_\sigma$, where P_σ is the policy set of $\mathcal{K} \restriction \sigma$.

Recall $\varphi = \psi_n$. By identifying, for every subset $X \subseteq \mathsf{A} \cup \overline{\mathsf{A}}$ the program X with $\bigcup_{x \in X} x$, we define

$$\varphi' \qquad = \qquad p_n \;\wedge\; [(\mathsf{A} \cup \overline{\mathsf{A}})^*] \bigwedge_{i \in [n]} (p_i \leftrightarrow \|\psi_i\|).$$

Note that expressing $\leftrightarrow$ with $\neg$ and $\vee$ only roughly doubles the size of the formula. The following upper bound on the size of φ' is not difficult to see.

Lemma 2. $|\varphi'| \in \mathcal{O}(|\varphi|^3)$.

The correctness of the above translation follows from the following lemma.

Lemma 3. *The formula φ is satisfiable if and only if φ' is satisfiable.*

By combining Lemma 2 and Lemma 3, we can deduce the following theorem.

Theorem 6. *There exists a satisfiability preserving translation from DLP$^{+}_{\mathrm{dyn}}$ to PDL$\oplus[\mathbb{A}]$, which is computable in polynomial time.*

6 Satisfiability for PDL$\oplus[\mathbb{A}]$

In this section, our goal is to prove that satisfiability of PDL$\oplus[\mathbb{A}]$ is in EXP. So for the rest of this section, fix some input PDL$\oplus[\mathbb{A}]$ formula φ over atomic propositions P and over atomic programs A. To decide satisfiability of φ, we translate φ into an alternating

---

$a \in \mathsf{A}$). However, the program $T^{\forall}(a)$ can be rewritten as

$$a^{M_\sigma} \cup \bigcup_{u \in U_\sigma} p(\theta_u)? \circ a^{M_\sigma \cap [u,k]} \circ p(\theta'_u)? \cup \bigcup_{u \in U_\sigma} p(\theta_u)? \circ \overline{a}^{M_\sigma \cap [u,k]} \circ p(\theta'_u)?$$

which is a PDL$\oplus[\mathbb{A}]$ program. We can proceed analogously for $T^{\exists}(a)$.

Büchi tree automaton $\mathcal{A}(\varphi)$ that accepts exactly the set of so called *Hintikka trees* for φ. Roughly speaking, Hintikka trees for φ summarize relevant information of models of φ. So the satisfiability of φ reduces to non-emptiness of the tree language of $\mathcal{A}(\varphi)$. We follow a similar approach as in [8]. Although in the following the $\neg$-operator may occur in front of non-atomic formulas, we implictly assume w.l.o.g. that every occurring PDL$\oplus[\mathbb{A}]$ formula is in *negation normal form*, i.e. negations occur only in front of atomic propositions. This can be achieved by introducing the operators $\wedge$ and $[\,]$ and by abbreviating $\neg(\varphi_1 \vee \varphi_2)$ by $\neg\varphi_1 \wedge \neg\varphi_2$ and $\neg\langle\pi\rangle\psi$ by $[\pi]\neg\psi$. Thus, we associate $\neg\neg\psi$ with ψ. We call formulas of the kind $\langle\pi\rangle\psi$ *diamond formulas* and formulas of the kind $[\pi]\psi$ *box formulas*. The definition of subf is straightforward. As for DLP$^{+}_{\mathrm{dyn}}$, for every program π that occurs in φ, we can associate a regular language $L(\pi)$ over the alphabet $\Sigma(\pi)$, where the latter consists of the basic programs and the test programs that occur in π. Moreover, we assume that the programs π, that occur in φ, are given by NFAs $A(\pi)$ over the alphabet $\Sigma(\pi)$, i.e. $L(A(\pi)) = L(\pi)$. Moreover, if $\mathcal{K}$ is a Kripke structure and $w = w_1 \cdots w_n$ with $w_i \in \Sigma(A(\pi))$ for all $i \in [n]$, then $[\![w]\!]_{\mathcal{K}} = [\![w_1]\!]_{\mathcal{K}} \circ \cdots \circ [\![w_n]\!]_{\mathcal{K}}$. If $A = (Q, \Sigma, q_0, \delta, F)$ is an NFA and $q \in Q$, then $A_q = (Q, \Sigma, q, \delta, F)$ denotes the same NFA as A, but with initial state q. If we do not explicitly define A, then $Q(A)$ denotes the state set of A, $\Sigma(A)$ denotes the alphabet of A, $q_0(A)$ denotes the initial state of A, and $F(A)$ denotes the set of final states of A respectively.
We start by introducing the *closure* of φ analogously as in [8, 14, 4].

Definition 1. *The* closure $\mathsf{cl}(\varphi)$ *of φ is the smallest set such that:*

- $\varphi \in \mathsf{cl}(\varphi)$,
- *if* $\chi \in \mathrm{subf}(\psi)$ *for some* $\psi \in \mathsf{cl}(\varphi)$*, then* $\chi \in \mathsf{cl}(\varphi)$,
- *if* $\psi \in \mathsf{cl}(\varphi)$*, then* $\neg\psi \in \mathsf{cl}(\varphi)$,
- *if* $\langle A\rangle\psi \in \mathsf{cl}(\varphi)$*, then* $\chi \in \mathsf{cl}(\varphi)$ *for all* $\chi? \in \Sigma(A)$,
- *if* $\langle A\rangle\psi \in \mathsf{cl}(\varphi)$, $\alpha \in \Sigma(A)$ *is a basic program, and* $\chi \in \mathrm{subf}(\alpha)$*, then* $\chi \in \mathsf{cl}(\varphi)$,
- *if* $\langle A\rangle\psi \in \mathsf{cl}(\varphi)$*, then for all* $q \in Q(A)$ *we have* $\langle A_q\rangle\psi \in \mathsf{cl}(\varphi)$,
- *if* $[A]\psi \in \mathsf{cl}(\varphi)$*, then* $\chi \in \mathsf{cl}(\varphi)$ *for all* $\chi? \in \Sigma(A)$,
- *if* $[A]\psi \in \mathsf{cl}(\varphi)$, $\alpha \in \Sigma(A)$ *is a basic program, and* $\chi \in \mathrm{subf}(\alpha)$*, then* $\chi \in \mathsf{cl}(\varphi)$*, and finally*
- *if* $[A]\psi \in \mathsf{cl}(\varphi)$*, then for all* $q \in Q(A)$ *we have* $[A_q]\psi \in \mathsf{cl}(\varphi)$.

It is straightforward to verify, that the size of $\mathsf{cl}(\varphi)$ is polynomial in the size of φ. Let us now introduce Hintikka sets for φ. These are subsets of $\mathsf{cl}(\varphi)$, that satisfy certain closure properties.

Definition 2. *A subset* $M \subseteq \mathsf{cl}(\varphi)$ *is a* Hintikka set for φ*, if the following five closure properties are satisfied:*

(C1) *if* $\chi_1 \wedge \chi_2 \in M$*, then* $\chi_1, \chi_2 \in M$,
(C2) *if* $\chi_1 \vee \chi_2 \in M$*, then* $\chi_1 \in M$ *or* $\chi_2 \in M$,
(C3) $\psi \in M$ *if and only if* $\neg\psi \notin M$ *for all* $\psi \in \mathsf{cl}(\varphi)$,
(C4) *if* $[A]\chi \in M$ *and* $q_0(A) \in F(A)$*, then* $\chi \in M$,
(C5) *if* $[A]\chi \in M$*, then for all* $q \in Q(A)$ *and all* $\chi'? \in \Sigma(A)$ *with* $q_0(A) \overset{\chi'?}{\Rightarrow}_A q$*, we have* $\neg\chi' \in M$ *or* $[A_q]\chi \in M$.

Recall that in any Hintikka set M for φ, by the presence of (C3), for all $\psi \in \mathsf{cl}(\varphi)$, we either have $\psi \in M$ or $\neg\psi \in M$ (but not both). Let $\mathcal{B}$ denote the set of all basic programs that occur in some diamond formula or in some box formula from $\mathsf{cl}(\varphi)$. For each $\alpha \in \mathcal{B}$ inductively define $\mathrm{At}(\alpha) = a$ if $\alpha = a \in \mathbb{A}$ and $\mathrm{At}(\alpha) = \mathrm{At}(\beta)$ if $\alpha = \oplus(\beta, \varphi_1, \varphi_2)$. For handling nestings of applied $\oplus$-operators in basic programs, we introduce, for basic programs $\alpha \in \mathcal{B}$, an appropriate notion of whether α holds between two subsets of $\mathsf{cl}(\varphi)$.

Definition 3. *For all $\alpha \in \mathcal{B}$ and all subsets $S, T \subseteq \mathsf{cl}(\varphi)$ for φ, we define the relation $(S, T) \models \alpha$ inductively by the syntactic structure of α as follows:*

- *$(S, T) \models a$ for all subsets $S, T \subseteq \mathsf{cl}(\varphi)$, and all $a \in \mathsf{A}$.*
- *If $\alpha = \oplus(\beta, \chi_1, \chi_2)$, then $(S, T) \models \alpha$ if and only if ($\chi_1 \in S$ or $\chi_2 \in T$) and $(S, T) \models \beta$.*

Let us introduce infinite trees and infinite paths in them. If Γ and Υ are sets, then a *Γ-labeled Υ-tree* is a mapping $\mathcal{T} : D \to \Gamma$ for some non-empty prefix-closed subset $D \subseteq \Upsilon^*$. An *infinite path of $\mathcal{T}$* is a mapping $\tau : \omega \to \Upsilon$ such that $\tau(1) \cdots \tau(n) \in D$ for all $n \geq 1$. If $k \in \omega$, then a *Γ-labeled k-tree* is a Γ-labeled $[k]$-tree $\mathcal{T} : D \to \Gamma$ such that $D = [k]^*$.
Fix an enumeration $\psi_1, \ldots, \psi_k$ of all diamond formulas in $\mathsf{cl}(\varphi)$. Let us now define a concrete labeling set Γ, that we will comment on below in more detail. Define the labeling set $\Gamma = 2^{\mathsf{cl}(\varphi)} \times (\mathcal{B} \uplus \{\perp\}) \times [0, k]$. Intuitively, each model $\mathcal{K}$ of φ corresponds to some Γ-labeled k-tree $\mathcal{T}$ that contains the necessary information about how diamond formulas are satisfied in $\mathcal{K}$. We can think of it as assigning each $u \in [k]^*$ some state x of $\mathcal{K}$. The first component of $\mathcal{T}(u)$ contains exactly the set of formulas of $\mathsf{cl}(\varphi)$ that hold in x. The second component of $\mathcal{T}(u)$ either contains the information by which witnessing basic program the state x was reached or whether this information is not important (then it equals $\perp$). If, on the one hand, the third component of $\mathcal{T}(u)$ contains the information $i \in [k]$, then u has the obligation, in its subtree, to show that the diamond formula ψ_i is true in x. If, on the other hand, the third component of $\mathcal{T}(u)$ equals 0, then u is a witness that some diamond formula $\psi_j = \langle A \rangle \chi \in \mathsf{cl}(\varphi)$ holds in some world $y \in X$ – more precisely, we have $(\mathcal{K}, x) \models \chi$ and $(y, x) \in [\![w]\!]_{\mathcal{K}}$ for some $w \in L(A)$. For every $\gamma \in \Gamma$ and every $j \in \{1, 2, 3\}$, let γ_j denote the j-th component of γ.
Before defining what a Hintikka tree is, let us first, for Γ-labeled k-trees, introduce a notion of local consistency of the labeling $\mathcal{T}(u) \in \Gamma$ of worlds $u \in [k]^*$. For a string w over some alphabet Σ let $\mathrm{Occ}(w) \subseteq \Sigma$ denote the set of all letters that occur in w.

Definition 4. *If $\mathcal{T} : [k]^* \to \Gamma$ is a Γ-labeled k-tree, and $u \in [k]^*$ is some world, we say that $\mathcal{T}$ is* locally consistent in u, *if the following two conditions hold:*

(L1) Whenever $\psi_i = \langle A \rangle \chi \in \mathcal{T}(u)_1$, then for some sequence of test programs $w \in (\Sigma(A) \backslash \mathcal{B})^*$ and some state $q' \in Q(A)$ such that $\theta? \in \mathrm{Occ}(w)$ implies $\theta \in \mathcal{T}(u)_1$, $q_0(A) \stackrel{w}{\Rightarrow}_A q'$, and at least one of the following two conditions holds:
(a)
- *$q' \in F(A)$,*
- *$\chi \in \mathcal{T}(u)_1$,*
- *$\mathcal{T}(ui)_2 = \perp$, and*
- *$\mathcal{T}(ui)_3 = 0$, or*

(b) there exists some basic program $\alpha \in \Sigma(A) \cap \mathcal{B}$ and some state $q \in Q(A)$ such that
   - $q' \overset{\alpha}{\Rightarrow}_A q$,
   - $\psi_j = \langle A_q \rangle \chi \in \mathcal{T}(ui)_1$,
   - $\mathcal{T}(ui)_2 = \alpha$,
   - $\mathcal{T}(ui)_3 = j$, *and*
   - $(\mathcal{T}(u)_1, \mathcal{T}(ui)_1) \models \alpha$.

(L2) Whenever $[A]\chi \in \mathcal{T}(u)_1$ and $q_0(A) \overset{\alpha}{\Rightarrow}_A q$ for some basic program $\alpha \in \Sigma(A) \cap \mathcal{B}$ and some $q \in Q(A)$, then the following implication holds for all $j \in [k]$ with $\mathcal{T}(uj)_2 = \beta \in \mathcal{B}$ and $\mathrm{At}(\alpha) = \mathrm{At}(\beta)$:

$$(\mathcal{T}(u)_1, \mathcal{T}(uj)_1) \models \alpha \qquad \Rightarrow \qquad [A_q]\chi \in \mathcal{T}(uj)_1$$

Let us summarize the intention of local consistency of a Γ-labeled k-tree in a world $u \in [k]^*$. As indicated above, condition (L1) ensures that whenever u obliges to prove some diamond formula ψ_i, then this proof is provided in the subtree of u: Either we directly prove that ψ_i holds in u ((L1)(a)), or we delay this proof by obliging an appropriate successor of u to prove an appropriate diamond formula ψ_j ((L1)(b)). Conditon (L2), on the other hand, ensures that if u obliges to prove some box formula, then all relevant successors of u must oblige to prove all relevant box formulas. We call a diamond formula ψ_i *infinitely delaying in a world* $u \in [k]^*$ of some Γ-labeled k-tree $\mathcal{T}$, if there exists an infinite path $\tau : \omega \to [k]$ such that $\tau(1) = i$ and $\mathcal{T}(u\tau(1)\cdots\tau(n))_3 = \tau(n+1)$ for all $n \geq 1$.

Definition 5. *A* Hintikka tree for φ *is a Γ-labeled k-tree $\mathcal{T}$ such that*

(H1) $\varphi \in \mathcal{T}(\varepsilon)_1$,
(H2) $T(u)_1$ is a Hintikka set for φ for all $u \in [k]^*$,
(H3) $\mathcal{T}$ is locally consistent in all $u \in [k]^*$, and
(H4) for all $u \in [k]^*$ and all $i \in [k]$ such that $\psi_i \in \mathcal{T}(u)_1$, the diamond formula ψ_i is not infinitely delaying in u.

The following lemma can be shown.

Lemma 4. *The formula φ is satisfiable if and only if there exists a Hintikka tree for φ.*

Let us introduce alternating Büchi tree automata over infinite trees. Our goal is to construct an alternating Büchi tree automaton $\mathcal{A}(\varphi)$ which accepts exactly the set of all Hintikka trees for φ. Thus, satisfiability of φ reduces to non-emptiness of the tree language of $\mathcal{A}(\varphi)$. If X is a set, let $\mathbb{B}^+(X)$ denote the set of all *positive boolean formulas* over the set X. An *alternating Büchi tree automaton* over Λ-labeled l-trees is a tuple $\mathcal{A} = (S, s_0, \delta, F)$, where (1) S is a finite set of *states*, (2) $s_0 \in S$ is the *initial state*, (3) $\delta : S \times \Lambda \to \mathbb{B}^+(S \times ([l] \cup \{\varepsilon\}))$ maps every pair $(s, \lambda) \in S \times \Lambda$ to a positive boolean formula $\delta(s, \lambda)$ over $(S \times ([l] \cup \{\varepsilon\}))$, and (4) $F \subseteq S$ is a set of *final states*. Let $\mathcal{T} : [l]^* \to \Lambda$ be a Λ-labeled l-tree. A *run* of $\mathcal{A}$ on $\mathcal{T}$ is an $S \times [l]^*$-labeled ω-tree $\mathcal{R} : D \to S \times [l]^*$ (i.e. D is a non-empty prefix-closed subset of ω^*), which satisfies the following: (1) $\mathcal{R}(\varepsilon) = (s_0, \varepsilon)$, and (2) for all $u \in D$ such that $\mathcal{R}(u) = (s, x)$ and $\delta(s, \mathcal{T}(x)) = \theta$ there exists some set $Y = \{(s_1, i_1), \ldots, (s_n, i_n)\} \subseteq$

$S \times ([l] \cup \{\varepsilon\})$ such that (i) Y satisfies the formula θ, and (ii) for all $1 \leq j \leq n$, we have $uj \in D$ and $\mathcal{R}(uj) = (s_j, xi_j)$. We call a run $\mathcal{R}$ of $\mathcal{A}$ on $\mathcal{T}$ *accepting*, if all infinite paths $\tau : \omega \to \omega$ of $\mathcal{R}$ satisfy the Büchi acceptance condition, i.e. $\{s \in S \mid \mathcal{R}(\tau(1) \cdots \tau(n)) \in (\{s\} \times [l]^*)$ for infinitely many $n \in \omega\} \cap F \neq \emptyset$. Let $L(\mathcal{A}) = \{\mathcal{T} \mid$ there exists an accepting run of $\mathcal{A}$ on $\mathcal{T}\}$ denote the *language* accepted by $\mathcal{A}$. From [13], the following complexity for non-emptiness of the language of alternating Büchi automata over infinite trees can be derived .

Theorem 7 ([13]). *The language non-emptiness for alternating Büchi tree automata $\mathcal{A} = (S, s_0, \delta, F)$ over Λ-labeled l-trees is decidable in time exponential in $|S|$ and l, and in time polynomial in $|\Lambda|$ and in $|\delta|$.*

Note that in several definitions, alternating Büchi tree automata are not allowed to use ε-transitions. It can easily be verified that Theorem 7 still holds, if we allow ε-transitions. By translating Definition 1, 2, 3, 4, and 5 into an alternating Büchi tree automaton, we can derive the following lemma.

Lemma 5. *There exists an alternating Büchi tree automaton $\mathcal{A} = \mathcal{A}(\varphi) = (S, s_0, \delta, F)$ over Γ-labeled k-trees that accepts exactly the set of Hintikka trees for φ. Moreover, the size of S is polynomial in $|\varphi|$ and the size of δ is exponential in $|\varphi|$.*

Note that the Büchi acceptance condition suffices to check whether some diamond formula is not infinitely delaying in some world. Clearly, we need alternation to check $(S, T) \models \alpha$ for subsets $S, T \subseteq \mathsf{cl}(\varphi)$ and $\alpha \in \mathcal{B}$ and thus for checking local consistency (H3). By Theorem 7 and Lemma 5, the existence of a Hintikka tree for φ can be decided in time exponential in $|\varphi|$. Thus, by applying Lemma 4, we get the following theorem.

Theorem 8. *Satisfiability for PDL$\oplus[\mathbb{A}]$ is in* EXP.

7 2EXP-completeness of PDL$\oplus$

In this section, we prove that satisfiability for PDL$\oplus$ is complete for 2EXP. Let us first introduce alternating Turing machines. An alternating Turing machine (ATM) is a tuple $\mathcal{M} = (Q, \Sigma, \Gamma, q_0, \delta, \Box)$, where (i) $Q = Q_{\text{acc}} \uplus Q_{\text{rej}} \uplus Q_\exists \uplus Q_\forall$ is a finite *set of states*, which is partitioned into *accepting* (Q_{acc}), *rejecting* (Q_{rej}), *existential* ($Q_\exists$), and *universal* ($Q_\forall$) states, (ii) Γ is a finite *tape alphabet*, (iii) $\Sigma \subseteq \Gamma$ is the *input alphabet*, (iv) $q_0 \in Q$ is the *initial state*, (v) $\Box \in \Gamma \setminus \Sigma$ is the *blank symbol*, and (vi) the mapping $\delta : (Q_\exists \cup Q_\forall) \times \Gamma \to \text{Moves} \times \text{Moves}$ with $\text{Moves} = Q \times \Gamma \times \{\leftarrow, \rightarrow\}$, assigns to every pair $(q, \gamma) \in (Q_\exists \cup Q_\forall) \times \Gamma$ a pair of moves. So we assume that a configuration in a current state from $Q_\exists \cup Q_\forall$ has exactly two successors. We call a configuration C of $\mathcal{M}$ in current state $q \in Q$ *accepting*, if either (i) $q \in Q_{\text{acc}}$,(ii) $q \in Q_\exists$ and there exists an accepting successor configuration of C, or (iii) $q \in Q_\forall$ and all successor configurations of C are accepting. Let $L(\mathcal{M}) = \{w \in \Sigma^* \mid$ configuration $q_0 w$ is accepting$\}$ denote the *language* of $\mathcal{M}$.

Theorem 9. *Satisfiability for PDL$\oplus$ is* 2EXP*-complete.*

Proof (sketch). The upper bound follows easily by a translation from PDL$\oplus$ into PDL with an exponential blowup in formula size (cf. Remark 1) and by the EXP upper bound of PDL (cf. Theorem 1). The proof of the lower bound is an adaption of the 2EXP lower bound for PDL with intersection from [6]. Fix some ATM $\mathcal{M} = (Q, \Sigma, \Gamma, q_0, \delta, \Box)$ with a 2EXP-hard acceptance problem, that, on an input $w = w_1 \cdots w_n \in \Sigma^*$ where $w_i \in \Sigma$ for each $i \in [n]$, uses at most $2^{p(n)}$ space, where p is some polynomial. By [1], such a Turing machine $\mathcal{M}$ exists. We will give a reduction, computable in polynomial time in n, that translates w and $\mathcal{M}$ into a PDL$\oplus$ formula $\varphi = \varphi(\mathcal{M}, w)$ such that $w \in L(\mathcal{M})$ if and only if φ is satisfiable. Let $N = p(n)$ and assume that Q and Γ are disjoint. A *configuration* of $\mathcal{M}$ is a word from the language $\bigcup_{i=0}^{2^N-1} \Gamma^i Q \Gamma^{2^N-i}$. Let $C = \gamma_0 \cdots \gamma_{i-1} q \gamma_i \cdots \gamma_{2^N-1}$ be a configuration of $\mathcal{M}$, where $0 \leq i \leq 2^N - 1$, $q \in Q$, and $\gamma_j \in \Gamma$ for each $0 \leq j \leq 2^N - 1$. Figure 1 illustrates how we will represent C.

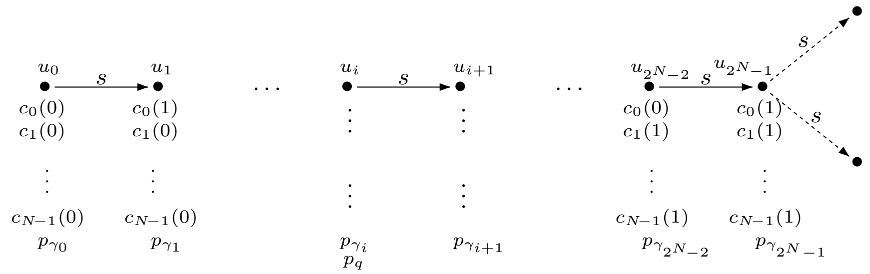

Fig. 1. Representation of a configuration of $\mathcal{M}$.

So each world u_j ($0 \leq j \leq 2^N - 1$) represents the cell on position j of the configuration C (starting to count from 0) and the symbols below each world are the atomic propositions that hold in u_j. The atomic propositions $c_l(0)$ and $c_l(1)$, where $0 \leq l \leq N-1$, represent the binary encoding of j. For each $0 \leq j \leq 2^N - 1$, there exists exactly one $\gamma \in \Gamma$ such that the atomic proposition p_γ holds in u_j, expressing that the content of the cell on position j of C is γ. On the other hand, for exactly one $0 \leq j < 2^N - 1$ and exactly one $q \in Q$, the atomic proposition p_q holds in u_j, expressing that $\mathcal{M}$ is currently in state q and scans cell j. Furthermore, worlds u_j and u_{j+1} are connected by the atomic program s for all $0 \leq j < 2^N - 1$. Moreover, the world u_{2^N-1} may be connected to the first cell of one or both successor configuration(s) of C via the atomic program s. Let $\Lambda = Q \uplus \Gamma$. Formally, the formula φ will be over the atomic propositions $\mathsf{P} = \{c_j(0), c_j(1) \mid 0 \leq j \leq N-1\} \cup \{p_\lambda \mid \lambda \in \Lambda\}$ and over the atomic program set $\mathsf{A} = \{s\}$. The crucial part of the formula φ is to find a program match that relates the current cell to the cell of some successor configuration at same positions. Let $\varphi_{\text{first}} = \bigwedge_{1 \leq j \leq N-1} c_j(0)$, which expresses that the current world represents some cell at position 0. The auxiliary program $\pi = (s \circ \neg\varphi_{\text{first}}?)^* \circ s \circ \varphi_{\text{first}}? \circ (s \circ \neg\varphi_{\text{first}}?)^*$ relates a cell c with a cell c' such that c' is in some successor configuration of the configuration of c, not necessarily at same positions. Let $\alpha_{-1} = \pi$ and, for all $0 \leq i \leq N-1$, define $\alpha_i = \oplus(\oplus(\alpha_{i-1}, c_i(0), c_i(1)), c_i(1), c_i(0))$. We put $\mathrm{match} = \alpha_{N-1}$. Since we enforced that each reachable world in a model of φ satisfies exactly one of the atomic propositions $c_i(0)$ and $c_i(1)$ for all $0 \leq i \leq N-1$, the program match relates only worlds that agree on the same atomic propositions from $\{c_i(0), c_i(1) \mid 0 \leq i \leq N-1\}$.

Since the program π relates cells in consecutive configurations, we relate cells in consecutive configurations at same positions. $\square$

Acknowledgments The author thanks the anonymous referees for interesting and substantial comments. Moreover the author thanks Markus Lohrey and Carsten Lutz for fruitful discussions on the topic as well as Florian Student for reading a draft version of this paper.

References

1. A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133, 1981.
2. R. Danecki. Nondeterministic Propositional Dynamic Logic with Intersection is decidable. In *Proc. of FCT 1984*, LNCS 208, pages 34–53, Springer 1984.
3. S. Demri. A reduction from DLP to PDL. *Journal of Logic and Computation*, 15(5):767–785, 2005.
4. M. J. Fischer and R. E. Ladner. Propositional Dynamic Logic of Regular Programs. *Journal of Computer and System Sciences*, 18(2):194-211, 1979.
5. S. Göller, M. Lohrey, and C. Lutz. PDL with Intersection and Converse is 2EXP-Complete. In *Proc. of FoSSaCS 2007*, LNCS 4423, pages 198–212, Springer 2007.
6. M. Lange and C. Lutz. 2-ExpTime Lower Bounds for Propositional Dynamic Logics with Intersection. *Journal of Symbolic Logic*, 70(4):1072–1086, 2005.
7. C. Löding and P. Rohde. Model Checking and Satisfiability for Sabotage Modal Logic. In *Proc. of FST&TCS 2003*, LNCS 2914, pages 302–313, Springer 2003.
8. C. Lutz and D. Walther. PDL with Negation of Atomic Programs. *Journal of Applied Non-Classical Logic*, 15(2):189–214, 2005.
9. V. R. Pratt. A Practical Decision Method for Propositional Dynamic Logic: Preliminary Report. In *Proc. of STOC 1980*, pages 326–337, ACM Press 1978.
10. R. Pucella and V. Weissman. Reasoning about Dynamic Policies. In *Proc. of FoSSaCS 2004*, LNCS 2987, pages 453–467, Springer 2004.
11. J. van Benthem. An Essay on Sabotage and Obstruction. In *Proc. of Mechanizing Mathematical Reasoning*, LNCS 2605, pages 268–276, Springer 2005.
12. R. van der Meyden. The Dynamic Logic of Permission. *Journal of Logic and Computation*, 6(3):465–479, 1996.
13. M. Y. Vardi. Reasoning about the Past with Two-Way Automata. In *Proc. of ICALP 98*, LNCS 1443, pages 628–641, Springer 1998.
14. M. Y. Vardi and P. Wolper. Automata-Theoretic Techniques for Modal Logics of Programs. *Journal of Computer and System Sciences*, 32(2):183–221, 1986.
15. R. J. Wieringa and J.-J. C. Meyer. Applications of Deontic Logic in Computer Science: A Concise Overview. In *Deontic Logic in Computer Science*, pages 17–40, 1993.

Appendix

Proof of Lemma 3

Proof.
$\Rightarrow$: Assume φ is satisfiable. Hence there exists an extended Kripke structure $\mathcal{K} = (X, \{\rightarrow_a \mid a \in \mathsf{A}\}, \varrho, P)$ over the atomic propositions P and over the atomic programs A s.t. for some world $x_0 \in X$ we have $(\mathcal{K}, x_0) \models \varphi$. By Lemma 3.3. in [3], we can assume that for all $a, b \in \mathsf{A}$ with $a \neq b$, we have $\rightarrow_a \cap \rightarrow_b = \emptyset$. Recall that $\{\psi_1, \ldots, \psi_n\}$ is an enumeration of $\mathrm{Occ}(\varphi)$ with $\varphi = \psi_n$. Define the Kripke structure $\mathcal{K}' = (X, \{\rightarrow'_a \mid a \in \mathsf{A} \cup \overline{\mathsf{A}}\}, \varrho')$ over the atomic propositions $\mathsf{P} \uplus \{p_1, \ldots, p_n\}$ and over the atomic programs $\mathsf{A} \uplus \overline{\mathsf{A}}$ as follows: For all $a \in \mathsf{A}$ define $\rightarrow'_a = \rightarrow_a \cap P$ and $\rightarrow'_{\overline{a}} = \rightarrow_a \setminus P$ and for all $x \in X$ we put $\varrho'(x) = \varrho(x) \cup \{p_i \mid 1 \leq i \leq n \wedge (\mathcal{K} \restriction \sigma(\psi_i), x) \models \psi_i\}$. Hence, for all $i \in [n]$ we have $(\mathcal{K}', x) \models p_i$ if and only if $(\mathcal{K}, \sigma(\psi_i)) \models \psi_i$. Now, the following claim holds:
Claim: For every $x \in X$ and for every $\psi_i \in \mathrm{Occ}(\varphi)$ we have:

$$(\mathcal{K} \restriction \sigma(\psi_i), x) \models \psi_i \qquad \Leftrightarrow \qquad (\mathcal{K}', x) \models \|\psi_i\|.$$

The claim is proven below. It remains to show that $\mathcal{K}'$ is a model of φ'. Due to the definition of ϱ' and the above claim, for every $x \in X$ and for every $\psi_i \in \mathrm{Occ}(\varphi)$, we have $(\mathcal{K}', x) \models p_i$ if and only if $(\mathcal{K}', x) \models \|\psi_i\|$. Since $(\mathcal{K}, x_0) \models \varphi$ and $\varphi = \psi_n$, we also have $(\mathcal{K}', x_0) \models p_n$ by the above claim. Thus, we can conclude

$$(\mathcal{K}', x_0) \qquad \models \qquad p_n \wedge [(\mathsf{A} \cup \overline{\mathsf{A}})^*] \bigwedge_{i \in [n]} (p_i \leftrightarrow \|\psi_i\|).$$

We prove the above claim directly by distinction of the structure of ψ_i.

- If ψ_i is atomic, i.e. $\psi_i = p$ for some $p \in \mathsf{P}$, then by definition of $\|\cdot\|$, we have $\|\psi_i\| = p$. Thus, we have

$$\begin{aligned}
(\mathcal{K} \restriction \sigma(\psi_i), x) \models p \quad &\overset{\text{Remark2}}{\Leftrightarrow} \quad (\mathcal{K}, x) \models p \\
&\Leftrightarrow \quad p \in \varrho(x) \\
&\Leftrightarrow \quad p \in \varrho'(x) \\
&\Leftrightarrow \quad (\mathcal{K}', x) \models p.
\end{aligned}$$

- If $\psi_i = \neg\psi_j$, then $(\mathcal{K} \restriction \sigma(\psi_i), x) \models \psi_i$ if and only if $(\mathcal{K} \restriction \sigma(\psi_j), x) \not\models \psi_j$. The latter is equivalent to $(\mathcal{K}', x) \not\models p_j$ by definition of ϱ'. As $\|\psi_i\| = \neg p_j$, we are done.
- If $\psi_i = \psi_j \vee \psi_k$, then

$$\begin{aligned}
(\mathcal{K} \restriction \sigma(\psi_i), x) \models \psi_i \quad &\Leftrightarrow \quad (\mathcal{K} \restriction \sigma(\psi_j), x) \models \psi_j \text{ or } (\mathcal{K} \restriction \sigma(\psi_k), x) \models \psi_k \\
&\Leftrightarrow \quad (\mathcal{K}', x) \models p_j \text{ or } (\mathcal{K}', x) \models p_k \\
&\Leftrightarrow \quad (\mathcal{K}', x) \models p_j \vee p_k.
\end{aligned}$$

- If $\psi_i = \texttt{grant}(\psi_j, \psi_k)\psi_l$, then

$$\begin{aligned}(\mathcal{K} \upharpoonright \sigma(\psi_i), x) \models \psi_i \quad &\Leftrightarrow \quad (((\mathcal{K} \upharpoonright \sigma(\psi_i)) \upharpoonright (\psi_j, \psi_k, \cup), x) \models \psi_l \\ &\Leftrightarrow \quad (\mathcal{K} \upharpoonright \sigma(\psi_l), x) \models \psi_l \\ &\Leftrightarrow \quad (\mathcal{K}', x) \models p_l.\end{aligned}$$

- If $\psi_i = \texttt{revoke}(\psi_j, \psi_k)\psi_l$, then

$$\begin{aligned}(\mathcal{K} \upharpoonright \sigma(\psi_i), x) \models \psi_i \quad &\Leftrightarrow \quad (((\mathcal{K} \upharpoonright \sigma(\psi_i)) \upharpoonright (\psi_j, \psi_k, \setminus), x) \models \psi_l \\ &\Leftrightarrow \quad (\mathcal{K} \upharpoonright \sigma(\psi_l), x) \models \psi_l \\ &\Leftrightarrow \quad (\mathcal{K}', x) \models p_l.\end{aligned}$$

- If $\psi_i = \langle\pi\rangle\psi_j$, then by definition of ϱ' for every test program $\psi_k?$ that occurs in π and for every $y \in X$ we have $(\mathcal{K} \upharpoonright \sigma(\psi_k), y) \models \psi_k$ if and only if $p_k \in \varrho'(y)$. Analogously, for every $y \in X$ we have $(\mathcal{K} \upharpoonright \sigma(\psi_j), y) \models \psi_j$ if and only if $p_j \in \varrho'(y)$ by definition of ϱ'. Now let $a \in \mathsf{A}$ be arbitrary. By Remark 2, we have $[\![a]\!]_{\mathcal{K}\upharpoonright\sigma(\psi_i)} = [\![a]\!]_{\mathcal{K}}$ and by construction of $\mathcal{K}'$ we have $[\![a]\!]_{\mathcal{K}} = [\![a]\!]_{\mathcal{K}'} \cup [\![\overline{a}]\!]_{\mathcal{K}'}$, hence $[\![a]\!]_{\mathcal{K}\upharpoonright\sigma(\psi_i)} = [\![a]\!]_{\mathcal{K}'} \cup [\![\overline{a}]\!]_{\mathcal{K}'} = [\![T(a)]\!]_{\mathcal{K}'}$. Since $T(\pi)$ is defined homomorphic on $\cup, \circ$, and on $*$, and $T(\psi_k?) = p_k?$ for test programs $\psi_k?$, it is easy to see that $(\mathcal{K} \upharpoonright \sigma(\psi_i), x) \models \langle\pi\rangle\psi_j$ if and only if $(\mathcal{K}', x) \models \langle T(\pi)\rangle p_j$.
- Assume $\psi_i = \texttt{perm}(\pi)\psi_j$, $\sigma = \sigma(\psi_i) = (\theta_1, \theta_1', \mathrm{op}_1)\cdots(\theta_k, \theta_k', \mathrm{op}_k)$ and $\mathcal{K} \upharpoonright \sigma = (X, \{\to_a \mid a \in \mathsf{A}\}, \varrho, P_\sigma)$. By definition of ϱ', for every $y \in X$ and for every $\psi_k \in \mathrm{Occ}(\varphi)$ such that either $\psi_k?$ occurs in π or ψ_k occurs in σ, we have $(\mathcal{K} \upharpoonright \sigma(\psi_k), y) \models \psi_k$ if and only if $p_k \in \varrho'(y)$. Analogously, for every $y \in X$ we have $(\mathcal{K} \upharpoonright \sigma(\psi_j), y) \models \psi_j$ if and only if $p_j \in \varrho'(y)$ by definition of ϱ'. Let $a \in \mathsf{A}$ be arbitrary. Again, by Remark 2, we have $[\![a]\!]_{\mathcal{K}\upharpoonright\sigma^{(l)}} = [\![a]\!]_{\mathcal{K}'} \cup [\![\overline{a}]\!]_{\mathcal{K}'}$ for all $l \in [0, k]$. Too, by construction of $\mathcal{K}'$, we have $[\![a]\!]_{\mathcal{K}} \cap P = [\![a]\!]_{\mathcal{K}'}$. Moreover, recall that for every subset $S = \{j_1, \ldots, j_l\} \subseteq [k]$ and every program ζ, we defined $\zeta^S = \zeta_l^S$, where $\zeta_0^S = \zeta$, and $\zeta_h^S = \oplus(\zeta_{h-1}^S, \neg p(\theta_{j_h}), \neg p(\theta_{j_h}'))$ for all $h \in [1, l]$. Taking this all together and by applying Lemma 1, we can formulate $(y, z) \in [\![a]\!]_{\mathcal{K}\upharpoonright\sigma} \cap P_\sigma$ as follows: There exists some $u \in \{0\} \uplus U_\sigma$ such that
$$(y, z) \in \begin{cases} [\![a^{M_\sigma}]\!]_{\mathcal{K}'} & \text{if } u = 0 \\ [\![p(\theta_u)? \circ (a \cup \overline{a})^{M_\sigma \cap [u,k]} \circ p(\theta_u)'?]\!]_{\mathcal{K}'} & \text{else} \end{cases}.$$
Hence, for all $(y, z) \in X \times X$ we have $(y, z) \in [\![a]\!]_{\mathcal{K}\upharpoonright\sigma} \cap P_\sigma$ if and only if $(y, z) \in [\![T^\forall(a)]\!]_{\mathcal{K}'}$. Since $T^\forall$ is defined homomorphic on $\cup, \circ$, and on $*$, and $T^\forall(\psi_k?) = p_k?$ for test programs $\psi_k?$, it follows that $(\mathcal{K} \upharpoonright \sigma(\psi_i), x) \models \psi_i)$ if and only if $(\mathcal{K}', x) \models \langle T^\forall(\pi)\rangle p_j$.
- $\psi_i = \texttt{fperm}(\pi)\psi_j$ can be proven analogously to the previous case.

$\Leftarrow$: Assume φ' is satisfiable. Hence, there exists an extended Kripke structure $\mathcal{K}' = (X, \{\to_a' \mid a \in \mathsf{A} \uplus \overline{\mathsf{A}}\}, \varrho)$ over the atomic propositions $\mathsf{P} \uplus \{p_1, \ldots, p_n\}$ and over the atomic programs $\mathsf{A} \uplus \overline{\mathsf{A}}$ such that for some $x_0 \in X$ we have $(\mathcal{K}', x_0) \models \varphi'$. Since PDL$\oplus[\mathbb{A}]$ is equally expressive as PDL and PDL is a fragment of DLP$^+_{\text{dyn}}$, we can again apply Lemma 3.3. in [3] and assume that for all $a, b \in \mathsf{A} \cup \overline{\mathsf{A}}$ with $a \neq b$ we have

$\to'_a \cap \to'_b = \emptyset$. Let $\mathcal{K} = (X, \{\to_a \mid a \in \mathsf{A}\}, \varrho, P)$ be the extended Kripke structure over the atomic propositions P and over the atomic programs A, where for all $a \in \mathsf{A}$ we have $\to_a = \to'_a \cup \to'_{\overline{a}}$, for all $x \in X$ we have $\varrho(x) = \varrho'(x) \cap \mathsf{P}$, and $P = \bigcup_{a \in \mathsf{A}} \to'_a$. Recall $\varphi = \psi_n$ and

$$\varphi' \qquad = \qquad p_n \ \wedge\ [(\mathsf{A} \cup \overline{\mathsf{A}})^*] \bigwedge_{i \in [n]} (p_i \leftrightarrow \|\psi_i\|).$$

We call a state $x \in X$ *reachable* from x_0 if $(x_0, x) \in \left(\bigcup_{a \in \mathsf{A}} \to'_a \cup \to'_{\overline{a}}\right)^*$ (which is equivalent to $(x_0, x) \in \left(\bigcup_{a \in \mathsf{A}} \to_a\right)^*$). Now, the following claim holds:
Claim: For every $\psi_i \in \mathrm{Occ}(\varphi)$ and for every $x \in X$ that is reachable from x_0, we have

$$(\mathcal{K} \restriction \sigma(\psi_i), x) \models \psi_i \qquad \Leftrightarrow \qquad (\mathcal{K}', x) \models p_i.$$

The claim is proven below. We show that $\mathcal{K}$ is a model for φ. By assumption, we have $(\mathcal{K}', x_0) \models \varphi'$, thus also $(\mathcal{K}', x_0) \models p_n$. Since $\psi_n = \varphi$ and by applying the above claim, we have $(\mathcal{K}, x_0) \models \varphi$. We proceed by proving the above claim. Recall that

$$(\mathcal{K}', x_0) \qquad \models \qquad p_n \ \wedge\ [(\mathsf{A} \cup \overline{\mathsf{A}})^*] \bigwedge_{i \in [n]} (p_i \leftrightarrow \|\psi_i\|).$$

Define the binary relation $< \subseteq \mathrm{Occ}(\varphi) \times \mathrm{Occ}(\varphi)$ as follows:

$$\psi_i < \psi_j \qquad \Leftrightarrow \qquad \psi_i \neq \psi_j \text{ and } (\psi_i \in \mathrm{Occ}(\psi_j) \text{ or } \psi_i \text{ occurs in } \sigma(\psi_j)).$$

Clearly, the relation $<$ is acyclic. Define $\sqsubset\, = \,<^+$. Hence, the relation $\sqsubset$ is a strict partial order on $\mathrm{Occ}(\varphi)$. We prove the above claim by induction on $\sqsubset$.
Base. Let ψ_i be minimal w.r.t. $\sqsubset$. Then $\psi_i = p$ for some $p \in \mathsf{P}$. Since $(\mathcal{K}', x_0) \models \varphi'$ and x is reachable from x_0, we have $(\mathcal{K}', x) \models p_i \leftrightarrow \|\psi_i\|$. Since $\|\psi_i\| = p$, we get

$$\begin{aligned}
(\mathcal{K} \restriction \sigma(\psi_i), x) \models p \quad &\overset{\text{Remark2}}{\Leftrightarrow} \quad (\mathcal{K}, x) \models p \\
&\Leftrightarrow \quad (\mathcal{K}', x) \models p \\
&\Leftrightarrow \quad (\mathcal{K}', x) \models p_i.
\end{aligned}$$

Step.

- If $\psi_i = \neg\psi_j$, then we have

$$\begin{aligned}
(\mathcal{K} \restriction \sigma(\psi_i), x) \models \psi_i \quad &\Leftrightarrow \quad (\mathcal{K} \restriction \sigma(\psi_i), x) \not\models \psi_j \\
&\Leftrightarrow \quad (\mathcal{K} \restriction \sigma(\psi_j), x) \not\models \psi_j \\
&\overset{\text{IH}}{\Leftrightarrow} \quad (\mathcal{K}', x) \not\models p_j \\
&\Leftrightarrow \quad (\mathcal{K}', x) \models \neg p_j \\
&\overset{(\mathcal{K}', x) \models p_i \leftrightarrow \neg p_j}{\Leftrightarrow} \quad (\mathcal{K}', x) \models p_i.
\end{aligned}$$

– If $\psi_i = \psi_j \vee \psi_k$, then we have

$$\begin{aligned}
(\mathcal{K} \upharpoonright \sigma(\psi_i), x) \models \psi_i \quad & \Leftrightarrow \quad (\mathcal{K} \upharpoonright \sigma(\psi_i), x) \models \psi_j \text{ or } (\mathcal{K} \upharpoonright \sigma(\psi_i), x) \models \psi_k \\
& \Leftrightarrow \quad (\mathcal{K} \upharpoonright \sigma(\psi_j), x) \models \psi_j \text{ or } (\mathcal{K} \upharpoonright \sigma(\psi_k), x) \models \psi_k \\
& \overset{\text{IH}}{\Leftrightarrow} \quad (\mathcal{K}', x) \models p_j \text{ or } (\mathcal{K}', x) \models p_k \\
& \Leftrightarrow \quad (\mathcal{K}', x) \models p_j \vee p_k \\
& \overset{(\mathcal{K}',x)\models p_i \leftrightarrow p_j \vee p_k}{\Leftrightarrow} \quad (\mathcal{K}', x) \models p_i .
\end{aligned}$$

– Assume $\psi_i = \langle \pi \rangle \psi_j$. By IH, for every state $y \in X$ that is reachable from x_0 we have (i) $(\mathcal{K} \upharpoonright \sigma(\psi_k), y) \models \psi_k$ if and only if $(\mathcal{K}', y) \models p_k$ for every $\psi_k \in \mathrm{Occ}(\varphi)$ such that either $\psi_k?$ occurs as a test program of π or ψ_k occurs in σ and (ii) $(\mathcal{K} \upharpoonright \sigma(\psi_j), y) \models \psi_j$ if and only if $(\mathcal{K}', y) \models p_j$. By definition of $\mathcal{K}$ and by Remark 2, we have $[\![a]\!]_{\mathcal{K}\upharpoonright\sigma} = [\![a]\!]_{\mathcal{K}} = [\![a]\!]_{\mathcal{K}'} \cup [\![\overline{a}]\!]_{\mathcal{K}'}$, thus $[\![a]\!]_{\mathcal{K}\upharpoonright\sigma} = [\![T(a)]\!]_{\mathcal{K}'}$. Furthermore, it is clear that whether $(\mathcal{K} \upharpoonright \sigma(\psi_i), x) \models \psi_i$ or not, depends only on those $y \in X$ that are reachable from x (and thus reachable from x_0). Since $T(\pi)$ is defined homomorphic on $\cup, \circ$ and on $*$, $T(\psi_k?) = p_k?$ for test programs $\psi_k?$, it is easy to see that $(\mathcal{K} \upharpoonright \sigma(\psi_i), x) \models \langle \pi \rangle \psi_j$ if and only if $(\mathcal{K}', x) \models \langle T(\pi) \rangle p_j$. Since x is reachable from x_0 and we have $(\mathcal{K}', x) \models (\langle T(\pi) \rangle p_j) \leftrightarrow p_i$, we are done.

– If $\psi_i = \mathtt{grant}(\psi_j, \psi_k)\psi_l$, then we have

$$\begin{aligned}
(\mathcal{K} \upharpoonright \sigma(\psi_i), x) \models \psi_i \quad & \Leftrightarrow \quad (\mathcal{K} \upharpoonright \sigma(\psi_l), x) \models \psi_l \\
& \overset{\text{IH}}{\Leftrightarrow} \quad (\mathcal{K}', x) \models p_l \\
& \overset{(\mathcal{K}',x)\models(p_i \leftrightarrow p_l)}{\Leftrightarrow} \quad (\mathcal{K}', x) \models p_i .
\end{aligned}$$

– If $\psi_i = \mathtt{revoke}(\psi_j, \psi_k)\psi_l$, then we have

$$\begin{aligned}
(\mathcal{K} \upharpoonright \sigma(\psi_i), x) \models \psi_i \quad & \Leftrightarrow \quad (\mathcal{K} \upharpoonright \sigma(\psi_l), x) \models \psi_l \\
& \overset{\text{IH}}{\Leftrightarrow} \quad (\mathcal{K}', x) \models p_l \\
& \overset{(\mathcal{K}',x)\models(p_i \leftrightarrow p_l)}{\Leftrightarrow} \quad (\mathcal{K}', x) \models p_i .
\end{aligned}$$

– If $\psi_i = \mathtt{perm}(\pi)\psi_l$, then we can prove $(\mathcal{K} \upharpoonright \sigma(\psi_i), x) \models \psi_i \Leftrightarrow (\mathcal{K}', x) \models p_i$ similarly as for the case $\mathtt{fperm}$ below.

– Assume $\psi_i = \mathtt{fperm}(\pi)\psi_l$, $\sigma = \sigma(\psi_i) = (\theta_1, \theta_1', \mathrm{op}_1) \cdots (\theta_k, \theta_k', \mathrm{op}_k)$ and $\mathcal{K} \upharpoonright \sigma = (X, \{\rightarrow_a \mid a \in \mathsf{A}\}, \varrho, P_\sigma)$. By IH, for every state $y \in X$ that is reachable from x_0 and every $\psi_k \in \mathrm{subf}(\varphi)$ such that either $\psi_k?$ occurs as a test program in π, or ψ_k occurs in σ, we have $(\mathcal{K} \upharpoonright \sigma(\psi_k), y) \models \psi_k$ if and only if $(\mathcal{K}', y) \models p_k$. Too, by IH, for all $y \in X$ that are reachable from x_0, we have $(\mathcal{K} \upharpoonright \sigma(\psi_j), y) \models \psi_j$ if and only if $(\mathcal{K}', y) \models p_j$. Let $a \in \mathsf{A}$ be arbitrary. By construction of $\mathcal{K} = (X, \{\rightarrow_a \mid a \in \mathsf{A}\}, \varrho, P)$ and by Remark 2, we have $[\![a]\!]_{\mathcal{K}} \setminus P = [\![\overline{a}]\!]_{\mathcal{K}'}$ and $[\![a]\!]_{\mathcal{K}\upharpoonright\sigma} = [\![a]\!]_{\mathcal{K}'} \cup [\![\overline{a}]\!]_{\mathcal{K}'} = [\![T(a)]\!]_{\mathcal{K}'}$. Moreover, recall that for every subset $S \subseteq \{j_1, \ldots, j_l\} \subseteq [k]$ and every program ζ, we have $\zeta^S = \zeta^S_l$, where $\zeta^S_0 = \zeta$ and $\zeta^S_h = \oplus(\zeta^S_{h-1}, \neg p(\theta_{j_h}), \neg p(\theta'_{j_h}))$ for all $h \in [1, l]$. Taking this all together and by applying Lemma 1, we can, for all $y, z \in X$ that are reachable from

x_0, formulate $(y,z) \in [\![a]\!]_{\mathcal{K}\restriction\sigma} \setminus P_\sigma$ as follows: There exists some $m \in \{0\} \uplus M_\sigma$ such that $(y,z) \in \begin{cases} [\![\overline{a}^{U_\sigma}]\!]_{\mathcal{K}'} & \text{if } m = 0 \\ [\![p(\theta_m)? \circ (a \cup \overline{a})^{U_\sigma \cap [m,k]} \circ p(\theta'_m)?]\!]_{\mathcal{K}'} & \text{else} \end{cases}$.
Hence, for all $y, z \in X$ that are reachable from x_0, we have $(y,z) \in [\![a]\!]_{\mathcal{K}\restriction\sigma} \setminus P_\sigma$ if and only if $(y,z) \in [\![T^\exists(a)]\!]_{\mathcal{K}'}$. Moreover, it is clear that, whether $(\mathcal{K} \restriction \sigma(\psi_i), x) \models \psi_i$ or not, depends only on those worlds that are reachable from x (and thus reachable from x_0). Hence, it is easy to see, by definition of $T^\exists(\pi)$, that we have $(\mathcal{K} \restriction \sigma(\psi_i), x) \models \texttt{fperm}(\pi)\psi_j$ if and only if $(\mathcal{K}', x) \models \neg\langle T^\exists(\pi)\rangle p_j$. Since x is reachable from x_0 and thus $(\mathcal{K}', x) \models (\neg\langle T^\exists(\pi)\rangle p_j) \leftrightarrow p_i$, we can conclude $(\mathcal{K}', x) \models p_i$.

□

Proof of Lemma 4

Proof.
$\Rightarrow$: Assume φ is satisfiable. Then, there exists a Kripke structure $\mathcal{K} = (X, \{\to_a \mid a \in \mathsf{A}\}, \varrho)$ and some world $z \in X$ with $(\mathcal{K}, z) \models \varphi$. Recall that $\psi_1, \ldots, \psi_k$ is an enumeration of all diamond formulas of $\mathsf{cl}(\varphi)$. For each NFA A that appears in some diamond formula $\langle A\rangle\chi \in \mathsf{cl}(\varphi)$, fix an arbitrary total order $\sqsubset_A$ on $\Sigma(A)$. We extend the total order $\sqsubset_A$ to words over $\Sigma(A)$ as the corresponding length lexicographic order w.r.t. $\sqsubset_A$, i.e. for all $u, v \in \Sigma(A)^*$ we define $u \sqsubset_A v$ if and only if either (i) $|u| < |v|$, or (ii) $|u| = |v|$ and for some $w \in \Sigma(A)^*$ and some $b, c \in \Sigma(A)$ with $b \sqsubset_A c$ we have that wb is a prefix of u and wc is a prefix of v. Let us, for each $x \in X$ and each $i \in [k]$ such that $\psi_i = \langle A\rangle\chi$ and $(\mathcal{K}, x) \models \psi_i$, fix the $\sqsubset_A$-minimal sequence $\sigma_{x,i} = b_1 \cdots b_n$ ($n \geq 0$), where $b_j \in \Sigma(A)$ for each $j \in [n]$, and fix some *involved worlds* $x_0, x_1, \ldots, x_n \in X$ and some *involved states* $q_0, q_1, \ldots, q_n \in Q(A)$ such that

- $x_0 = x$,
- $q_0 = q_0(A)$,
- $(x_{j-1}, x_j) \in [\![b_j]\!]_{\mathcal{K}}$ for all $j \in [n]$,
- $q_{j-1} \overset{b_j}{\Rightarrow}_A q_j$ for all $j \in [n]$,
- $(\mathcal{K}, x_n) \models \chi$, and
- $q_n \in F(A)$.

Recall that $\mathcal{B}$ denotes the set of all basic programs that occur in φ and $\Gamma = 2^{\mathsf{cl}(\varphi)} \times (\mathcal{B} \uplus \{\bot\}) \times [0,k]$. Our goal is to find a Hintikka tree for φ. For this, inductively, by the length of the elements from $[k]^*$, we simultaneously define a Γ-labeled k-tree $\mathcal{T} : [k]^* \to \Gamma$ and a mapping $\Psi : [k]^* \to X$. We set

$$\begin{aligned} \Psi(\varepsilon) &= z, \\ \mathcal{T}(\varepsilon)_1 &= \{\psi \in \mathsf{cl}(\varphi) \mid (\mathcal{K}, z) \models \psi\}, \\ \mathcal{T}(\varepsilon)_2 &= \bot, \text{ and} \\ \mathcal{T}(\varepsilon)_3 &= 0. \end{aligned}$$

The definition of $\mathcal{T}$ and Ψ will have the property that for all $u \in [k]^*$ we have

$$\mathcal{T}(u)_1 \quad = \quad \{\psi \in \mathsf{cl}(\varphi) \mid (\mathcal{K}, \Psi(u)) \models \psi\}. \tag{1}$$

Now assume that $\Psi(u)$ and $\mathcal{T}(u)$ have been defined for $u \in [k]^*$ but yet $\Psi(ui)$ and $\mathcal{T}(ui)$ have not been defined for all $i \in [k]$. Assume $\Psi(u) = x$ for some $x \in X$. Then, for each $i \in [k]$, we do the following:

- If $\psi_i = \langle A \rangle \chi \in T(u)_1$, then we know, by equation (1), that $(\mathcal{K}, x) \models \psi_i$. We distinguish two cases:
  - $\sigma_{x,i}$ is either empty or only consists of test programs. Then, we put

$$\begin{aligned} \Psi(ui) &= z, \\ \mathcal{T}(ui)_1 &= \{\psi \in \mathsf{cl}(\varphi) \mid (\mathcal{K}, z) \models \psi\}, \\ \mathcal{T}(ui)_2 &= \bot, \text{ and} \\ \mathcal{T}(ui)_3 &= 0. \end{aligned}$$

  - $\sigma_{x,i} = b_1 \cdots b_n$ contains at least one basic program. Moreover, let $q_0, q_1, \ldots, q_n \in Q(A)$ be the involved states and let $x_0, x_1, \ldots, x_n \in X$ be the involved worlds for $\sigma_{x,i}$. Let $l \in [n]$ be minimal such that $b_l = \alpha \in \mathcal{B}$ is a basic program and let $\psi_j = \langle A_{q_l} \rangle \chi$. Then, we put

$$\begin{aligned} \Psi(ui) &= x_l, \\ \mathcal{T}(ui)_1 &= \{\psi \in \mathsf{cl}(\varphi) \mid (\mathcal{K}, x_l) \models \psi\}, \\ \mathcal{T}(ui)_2 &= \alpha, \text{ and} \\ \mathcal{T}(ui)_3 &= j. \end{aligned}$$

- If $\psi_i = \langle A \rangle \chi \notin \mathcal{T}(u)_1$, then we put

$$\begin{aligned} \Psi(ui) &= z, \\ \mathcal{T}(ui)_1 &= \{\psi \in \mathsf{cl}(\varphi) \mid (\mathcal{K}, z) \models \psi\}, \\ \mathcal{T}(ui)_2 &= \bot, \text{ and} \\ \mathcal{T}(ui)_3 &= 0. \end{aligned}$$

Before proving that $\mathcal{T}$ is indeed a Hintikka tree, we prove the following claim:
Claim: For all $u \in [k]^*$ and all $i \in [k]$ such that $\mathcal{T}(ui)_2 = \beta$ for some $\beta \in \mathcal{B}$, the following equivalence holds for all $\alpha \in \mathcal{B}$ with $\mathrm{At}(\alpha) = \mathrm{At}(\beta)$:

$$(\Psi(u), \Psi(ui)) \in [\![\alpha]\!]_{\mathcal{K}} \quad \Leftrightarrow \quad (\mathcal{T}(u)_1, \mathcal{T}(ui)_1) \models \alpha.$$

Proof (of claim). Let $\mathrm{At}(\alpha) = \mathrm{At}(\beta) = a$ for some $a \in \mathsf{A}$. We prove the claim by induction on the syntactic structure of α, which is possible since $\mathrm{At}(\alpha') = \mathrm{At}(\alpha)$ for all basic programs α' that occur in α.
*B*ase. Assume α is atomic. Since $\mathcal{T}(ui)_2 = \beta$, we have $(\Psi(u), \Psi(ui)) \in [\![\beta]\!]_{\mathcal{K}}$ by definition of $\mathcal{T}$. Since α is atomic, it is clear that also $(\Psi(u), \Psi(ui)) \in [\![\alpha]\!]_{\mathcal{K}}$. On the other hand, directly by definition of $\models$, we have $(\mathcal{T}(u)_1, \mathcal{T}(ui)_1) \models \alpha$.

Step. Now assume $\alpha = \oplus(\alpha', \chi_1, \chi_2)$ and, by IH, the claim already holds for α'. We have

$$
\begin{aligned}
& (\Psi(u), \Psi(ui)) \in [\![\alpha]\!]_{\mathcal{K}} \\
\Leftrightarrow \quad & (\Psi(u), \Psi(ui)) \in \oplus([\![\alpha']\!]_{\mathcal{K}}, [\![\chi_1]\!]_{\mathcal{K}}, [\![\chi_2]\!]_{\mathcal{K}}) \\
\Leftrightarrow \quad & (\Psi(u), \Psi(ui) \in [\![\alpha']\!]_{\mathcal{K}} \text{ and } (\Psi(u) \in [\![\chi_1]\!]_{\mathcal{K}} \text{ or } \Psi(ui) \in [\![\chi_2]\!]_{\mathcal{K}}) \\
\overset{\text{IH}}{\Leftrightarrow} \quad & (\mathcal{T}(u)_1, \mathcal{T}(ui)_1) \models \alpha' \text{ and } (\Psi(u) \in [\![\chi_1]\!]_{\mathcal{K}} \text{ or } \Psi(ui) \in [\![\chi_2]\!]_{\mathcal{K}}) \\
\overset{\text{eq. (1)}}{\Leftrightarrow} \quad & (\mathcal{T}(u)_1, \mathcal{T}(ui)_1) \models \alpha' \text{ and } (\chi_1 \in \mathcal{T}(u)_1 \text{ or } \chi_2 \in \mathcal{T}(ui)_1) \\
\overset{\text{Def. of} \models}{\Leftrightarrow} \quad & (\mathcal{T}(u)_1, \mathcal{T}(ui)_1) \models \alpha.
\end{aligned}
$$

This completes the proof of the above claim.

It remains to prove that $\mathcal{T}$ is indeed a Hintikka tree for φ.
Point (H1), i.e. $\varphi \in \mathcal{T}(\varepsilon)_1$, follows directly from the induction base of the definition of $\mathcal{T}$.

It follows directly by the semantics of PDL$\oplus[\mathbb{A}]$ that (H2) holds, i.e. $T(u)_1$ is a Hintikka set for φ for every $u \in [k]^*$.

Let us prove (H3), i.e. $\mathcal{T}$ is locally consistent in all $u \in [k]^*$. Let $\Psi(u) = x$ for some $x \in X$.
First, let us prove that (L1) holds. So let $i \in [k]$ and assume $\psi_i = \langle A \rangle \chi \in \mathcal{T}(u)_1$. Then, by equation (1), we have $(\mathcal{K}, x) \models \psi_i$. Assume $\sigma_{x,i} = b_1 \cdots b_n$, let $q_0, q_1, \ldots, q_n \in Q(A)$ be its involved states, and let $x_0, x_1, \ldots, x_n \in X$ be its involved worlds. Furthermore, let $w = \chi_1? \cdots \chi_l?$ ($l \geq 0$) be the longest prefix of $\sigma_{x,i}$ that only consists of test programs. Thus, we have $q_0(A) \overset{w}{\Rightarrow}_A q_l$ and $(\mathcal{K}, x) \models \chi_j$ for all $j \in [l]$. Hence, $\{\chi_1, \ldots, \chi_l\} \subseteq \mathcal{T}(u)_1$ by equation (1). We distinguish the following two cases:

1. If $\sigma_{x,i}$ is empty or only consists of test programs, then $l = n$, $x_l = x$, and $q_l \in F(A)$. Since the latter implies $(\mathcal{K}, x) \models \chi$, we get $\chi \in \mathcal{T}(u)_1$ by equation (1). By construction of $\mathcal{T}$, we have $\mathcal{T}(ui)_2 = \perp$ and $\mathcal{T}(ui)_3 = 0$. Altogether, this implies (L1) (a).
2. Assume $\sigma_{x,i}$ contains at least one basic program. This implies $b_{l+1} = \alpha$ for some $\alpha \in \mathcal{B}$ and we have $q_l \overset{\alpha}{\Rightarrow}_A q_{l+1}$. Let $\psi_j = \langle A_{q_{l+1}} \rangle \chi$. Clearly, we have $(\mathcal{K}, x_{l+1}) \models \psi_j$ and due to construction of $\mathcal{T}$, we have $\Psi(ui) = x_{l+1}$. Thus, equation (1) yields $\psi_j \in \mathcal{T}(ui)_1$. By construction of $\mathcal{T}$, we also have $\mathcal{T}(ui)_2 = \alpha$ and $\mathcal{T}(ui)_3 = j$. Since $(\Psi(u), \Psi(ui)) \in [\![\alpha]\!]_{\mathcal{K}}$ and $\mathcal{T}(ui)_2 = \alpha$, the above claim implies $(\mathcal{T}(u)_1, \mathcal{T}(ui)_1) \models \alpha$. Altogether, this implies (L1) (b).

Let us now prove that (L2) holds. So assume $[A]\chi \in \mathcal{T}(u)_1$ and $q_0(A) \overset{\alpha}{\Rightarrow} q$ for some $\alpha \in \Sigma(A) \cap \mathcal{B}$ and some $q \in Q(A)$. Furthermore, assume, by contradiction, that for some $j \in [k]$ with $\mathcal{T}(uj)_2 = \beta$, where $\beta \in \mathcal{B}$ and $\mathrm{At}(\alpha) = \mathrm{At}(\beta)$, we have $(\mathcal{T}(u)_1, \mathcal{T}(uj)_1) \models \alpha$ and $[A_q]\chi \notin \mathcal{T}(uj)_1$. By equation (1), $[A_q]\chi \notin \mathcal{T}(uj)_1$ is equivalent to $(\mathcal{K}, \Psi(uj)) \models \langle A_q \rangle \neg\chi$. Since $(\mathcal{T}(u)_1, \mathcal{T}(uj)_1) \models \alpha$, we have $(\Psi(u), \Psi(uj)) \in [\![\alpha]\!]_{\mathcal{K}}$ by the above claim. By combining $q_0(A) \overset{\alpha}{\Rightarrow}_A q$ and $(\Psi(u), \Psi(uj)) \in [\![a]\!]_{\mathcal{K}}$ and $(\mathcal{K}, \Psi(uj)) \models \langle A_q \rangle \neg\chi$, we can deduce $(\mathcal{K}, \Psi(u)) \models \langle A \rangle \neg\chi$, which contradicts to $(\mathcal{K}, \Psi(u)) \models [A]\chi$ and thus also to $[A]\chi \in \mathcal{T}(u)_1$ by equation (1).

Before finally proving (H4), the following proposition holds:
Proposition: For all $u \in [k]^*$ and all $i \in [k]$ such that $\psi_i \in \mathcal{T}(u)_1$, $\Psi(u) = x$, $\Psi(ui) = y$, and $\mathcal{T}(ui)_3 = j$ for some $x, y \in X$ and some $j \in [k]$, we have

$$|\sigma_{x,i}| \qquad > \qquad |\sigma_{y,j}|.$$

The proposition follows directly by construction of $\mathcal{T}$ and by definition of $\sigma_{x,i}$ and $\sigma_{y,j}$. To prove (H4), assume, by contradiction, that for some $i \in [k]$ the diamond formula ψ_i is infinitely delaying in some $u \in [k]^*$. Furthermore, assume $\Psi(u) = x$ for some $x \in X$. But by the above proposition, the existence of an infinite path $\tau : \omega \to [k]$ with $\tau(1) = i$ and $\mathcal{T}(u\tau(1)\cdots\tau(n))_3 = \tau(n+1)$ for all $n \geq 1$ clearly contradicts the finiteness of $|\sigma_{x,i}|$.

$\Leftarrow$: Let $\mathcal{T} : [k]^* \to \Gamma$ be a Hintikka tree for φ. We will prove that there exists a model for φ. Define the Kripke structure $\mathcal{K} = (X, \{\to_a \mid a \in \mathsf{A}\}, \varrho)$ as follows:

- $X = [k]^*$,
- for all $a \in \mathsf{A}$, we define

  $$\to_a \qquad = \qquad \{(u, ui) \in [k]^* \times [k]^* \mid i \in [k], \mathcal{T}(ui)_2 = \alpha \in \mathcal{B} \text{ and } \mathrm{At}(\alpha) = a\},$$

- and for all $u \in [k]^*$, we define $\varrho(u) = \mathcal{T}(u)_1 \cap \mathsf{P}$.

Let us define the acyclic binary relation $\prec \subseteq \mathsf{cl}(\varphi) \times \mathsf{cl}(\varphi)$ as the smallest binary relation on $\mathsf{cl}(\varphi)$ that satisfies the following conditions:

- If $\psi = \chi_1 \wedge \chi_2 \in \mathsf{cl}(\varphi)$ or $\psi = \chi_1 \vee \chi_2 \in \mathsf{cl}(\varphi)$, then $\chi_i \prec \psi$ and $\neg\chi_i \prec \psi$ for all $i \in \{1, 2\}$.
- If $\psi = \langle A \rangle \chi \in \mathsf{cl}(\varphi)$ or $\psi = [A]\chi \in \mathsf{cl}(\varphi)$, then $\chi \prec \psi$, $\neg\chi \prec \psi$ and $\chi' \prec \psi$,$\neg\chi' \prec \psi$ for all formulas $\chi' \in \mathsf{cl}(\varphi)$ such that either $\chi'? \in \Sigma(A) \setminus \mathcal{B}$ is a test program or χ' occurs in some basic program of $\Sigma(A)$.

Let $\sqsubset = \prec^+$ be the transitive closure of $\prec$, which is a strict partial order on $\mathsf{cl}(\varphi)$. Note that the minimal elements of $\mathsf{cl}(\varphi)$ w.r.t. $\sqsubset$ are formulas ψ such that either $\psi = p$ or $\psi = \neg p$ for some $p \in \mathsf{P}$. Our goal is to prove, that, for all $\psi \in \mathsf{cl}(\varphi)$ and for all $u \in [k]^*$, the following implication holds:

$$\psi \in \mathcal{T}(u)_1 \qquad \Rightarrow \qquad (\mathcal{K}, u) \models \psi. \tag{2}$$

The lemma then follows immediateley, since, by (H1), we know $\varphi \in \mathcal{T}(\varepsilon)_1$ and thus, by implication (2), we get $(\mathcal{K}, \varepsilon) \models \varphi$. We prove implication (2) by induction on $\sqsubset$.
Base. Assume ψ is minimal w.r.t. $\sqsubset$. Hence $\psi = p$ or $\psi = \neg p$ for some $p \in \mathsf{P}$. It follows directly by the definition of ϱ and by (C3) that $\psi \in \mathcal{T}(u)_1$ implies $(\mathcal{K}, u) \models \psi$.
Step.

- If $\psi = \chi_1 \wedge \chi_2$ or $\psi = \chi_1 \vee \chi_2$, then $\psi \in \mathcal{T}(u)_1 \Rightarrow (\mathcal{K}, u) \models \psi$ follows immediateley by (C1),(C2), and by IH.
- Let $\psi = \langle A \rangle \chi$ or $\psi = [A]\chi$. Let us first prove the following claim:
  Claim: Let $u \in [k]^*$ and $i \in [k]$ and assume $(u, ui) \in \to_a$ for some $a \in \mathsf{A}$. Then, for all basic programs $\alpha \in \Sigma(A) \cap \mathcal{B}$ such that $\mathrm{At}(\alpha) = a$, we have

  $$(\mathcal{T}(u)_1, \mathcal{T}(ui)_1) \models \alpha \qquad \Leftrightarrow \qquad (u, ui) \in [\![\alpha]\!]_{\mathcal{K}}.$$

Proof (of claim). Since $\mathrm{At}(\alpha') = \mathrm{At}(\alpha)$ for all basic programs α' that occur in α, we can prove the claim by induction on the syntactic structure of α.
Base. Assume α is atomic. By definition of $\models$, we have $(\mathcal{T}(u)_1, \mathcal{T}(ui)_1) \models \alpha$. By assumption, we also have $(u, ui) \in [\![\alpha]\!]_{\mathcal{K}}$.
Step. Let $\alpha = \oplus(\alpha', \chi_1, \chi_2)$ and assume the claim is true for α'. Let us first prove that the following two statements are equivalent:
(i) $\chi_1 \in \mathcal{T}(u)_1$ or $\chi_2 \in \mathcal{T}(ui)_1$.
(ii) $u \in [\![\chi_1]\!]_{\mathcal{K}}$ or $ui \in [\![\chi_2]\!]_{\mathcal{K}}$.
The implication from (i) to (ii) follows directly by (C2) and the outer IH. To prove that (ii) implies (i), assume by contradiction and w.l.o.g., $u \in [\![\chi_1]\!]_{\mathcal{K}}$ and $\chi_1 \notin \mathcal{T}(u)_1$. Then, by (C3), we have $\neg\chi_1 \in \mathcal{T}(u)_1$. By the outer IH, we have $(\mathcal{K}, u) \not\models \chi_1$, which contradicts $u \in [\![\chi_1]\!]_{\mathcal{K}}$. Thus, we get

$$\begin{array}{cl}
& (\mathcal{T}(u)_1, \mathcal{T}(ui)_1) \models \alpha \\
\overset{\text{Def. of} \models}{\Leftrightarrow} & (\mathcal{T}(u)_1, \mathcal{T}(ui)_1) \models \alpha' \text{ and } (\chi_1 \in \mathcal{T}(u)_1 \text{ or } \chi_2 \in \mathcal{T}(ui)_1) \\
\overset{\text{(i)}\Leftrightarrow\text{(ii)}}{\Leftrightarrow} & (\mathcal{T}(u)_1, \mathcal{T}(ui)_1) \models \alpha' \text{ and } (u \in [\![\chi_1]\!]_{\mathcal{K}} \text{ or } ui \in [\![\chi_2]\!]_{\mathcal{K}}) \\
\overset{\text{inner IH}}{\Leftrightarrow} & (u, ui) \in [\![\alpha']\!]_{\mathcal{K}} \text{ and } (u \in [\![\chi_1]\!]_{\mathcal{K}} \text{ or } ui \in [\![\chi_2]\!]_{\mathcal{K}}) \\
\Leftrightarrow & (u, ui) \in \oplus([\![\alpha']\!]_{\mathcal{K}}, [\![\chi_1]\!]_{\mathcal{K}}, [\![\chi_2]\!]_{\mathcal{K}}) \\
\Leftrightarrow & (u, ui) \in [\![\alpha]\!]_{\mathcal{K}}.
\end{array}$$

This completes the proof of the claim.

Let us first prove that if $\psi = \langle A \rangle \chi$, then $\psi \in \mathcal{T}(u)_1$ implies $(\mathcal{K}, u) \models \psi$. Let $\psi = \psi_i$ for some $i \in [k]$ and assume that $\psi_i \in \mathcal{T}(u)_1$. Since $\mathcal{T}$ is a Hintikka tree for φ, and thus the diamond formula ψ_i is not infinitely delaying in u, there exists a unique finite sequence $\tau = j_1 \cdots j_n \in [k]^*$, $n \geq 1$, such that $j_1 = i$, $\mathcal{T}(u\tau)_3 = 0$ and $\mathcal{T}(u\tau^{(m)})_3 = j_{m+1}$ for all $1 \leq m < n$. Let $\Sigma_? = \Sigma(A) \setminus \mathcal{B}$ denote the set of all test programs that occur in $\Sigma(A)$. Since $\mathcal{T}(u\tau^{(m)})_3 = j_{m+1} \neq 0$ and $\mathcal{T}$ is locally consistent in $u\tau^{(m)}$ for all $1 \leq m < n$, condition (L1)(b) holds for all $u\tau^{(m)}$ with $1 \leq m < n$. On the other hand, since $\mathcal{T}(u\tau)_3 = 0$ and $\mathcal{T}$ is also locally consistent in $u\tau$, we know that condition (L1)(a) holds in $u\tau$. Taking these facts together, we conclude that there exist states $q_1, q_1', \ldots, q_n, q_n' \in Q(A)$, words $w_1, \ldots, w_{n-1} \in \Sigma_?^*$, and basic programs $\alpha_1, \ldots, \alpha_{n-1} \in \Sigma(A) \cap \mathcal{B}$ such that the following holds:
(1) $q_1 = q_0(A)$,
(2) $\mathrm{Occ}(w_m) \subseteq \mathcal{T}(u\tau^{(m-1)})_1$ for all $1 \leq m \leq n$,
(3) $q_m \overset{w_m}{\Rightarrow}_A q_m'$ for all $1 \leq m \leq n$,
(4) $q_m' \overset{\alpha_m}{\Rightarrow}_A q_{m+1}$ for all $1 \leq m < n$,
(5) $\psi_{j_m} = \langle A_{q_m} \rangle \chi \in \mathcal{T}(u\tau^{(m-1)})_1$ for all $1 \leq m < n$,
(6) $(\mathcal{T}(u\tau^{(m-1)})_1, \mathcal{T}(u\tau^{(m)})_1) \models \alpha_m$ for all $1 \leq m < n$,
(7) $q_n' \in F(A)$,
(8) $\chi \in \mathcal{T}(u\tau^{(n-1)})_1$, and
(9) $\mathcal{T}(u\tau)_2 = \bot$.
Taking the transitions from (3) and (4) together, we obtain the following run:

$$q_0(A) = q_1 \overset{w_1}{\Rightarrow}_A q_1' \overset{\alpha_1}{\Rightarrow}_A q_2 \cdots \overset{\alpha_{n-1}}{\Rightarrow}_A q_n \overset{w_n}{\Rightarrow}_A q_n'$$

Moreover, by the above claim, point (6) implies $(u\tau^{(m-1)}, u\tau^{(m)}) \in [\![\alpha_m]\!]_{\mathcal{K}}$ for all $1 \le m < n$. By the IH, point (2) implies $(u\tau^{(m-1)}, u\tau^{(m-1)}) \in [\![w_m]\!]_{\mathcal{K}}$ for all $1 \le m \le n$. Altogether, we obtain $(u, u\tau^{(n-1)}) \in [\![w_1\alpha_1 \cdots w_{n-1}\alpha_{n-1}w_n]\!]_{\mathcal{K}}$. Moreover, point (8) and the IH implies $(\mathcal{K}, u\tau^{(n-1)}) \models \chi$. Since, by point (1), $q_1 = q_0(A)$ and, by (7), $q'_n \in F(A)$, we conclude $(\mathcal{K}, u) \models \langle A \rangle \chi$.

It remains to prove that $[A]\chi \in \mathcal{T}(u)_1$ implies $(\mathcal{K}, u) \models [A]\chi$. For this, assume $[A]\chi \in \mathcal{T}(u)_1$. Let $u_0, u_1, \ldots, u_n \in [k]^*$ be worlds such that for some states $q_0, \ldots, q_n \in Q(A)$ and some $\sigma_1, \ldots, \sigma_n \in \Sigma(A)$ the following holds:
(1) $u_0 = u$,
(2) $q_0 = q_0(A)$,
(3) for all $1 \le j \le n$ we have $q_{j-1} \overset{\sigma_j}{\Rightarrow}_A q_j$,
(4) for all $1 \le j \le n$ we have $(u_{j-1}, u_j) \in [\![\sigma_j]\!]_{\mathcal{K}}$, and
(5) $q_n \in F(A)$.
We prove $(\mathcal{K}, u_n) \models \chi$. It suffices to prove that for all $0 \le j \le n$ we have $[A_{q_j}]\chi \in \mathcal{T}(u_j)_1$, since $q_n \in F(A)$ and $[A_{q_n}]\chi \in \mathcal{T}(u_n)_1$ implies $\chi \in \mathcal{T}(u_n)_1$ by (C4) and this yields $(\mathcal{K}, u_n) \models \chi$ by the outer IH. We prove $[A_{q_j}]\chi \in \mathcal{T}(u_j)_1$ by induction on j. The induction base, i.e. $[A_{q_0}]\chi \in \mathcal{T}(u_0)_1$, holds by assumption. For the induction step, let $0 \le j < n$ and assume $[A_{q_j}]\chi \in \mathcal{T}(u_j)_1$ already holds. To prove $[A_{q_{j+1}}]\chi \in \mathcal{T}(u_{j+1})_1$, we distinguish the following two cases:

- σ_{j+1} is a test program, i.e. $\sigma_{j+1} = \chi'?$ for some $\chi'? \in \Sigma(A) \setminus \mathcal{B}$. Hence, $u_j = u_{j+1}$ and $(\mathcal{K}, u_{j+1}) \models \chi'$. The outer IH and (C3) implies $\neg\chi' \notin \mathcal{T}(u_{j+1})_1$. Since also $q_0(A_{q_j}) = q_j \overset{\chi'}{\Rightarrow}_A q_{j+1}$, we get $[A_{q_{j+1}}]\chi \in \mathcal{T}(u_{j+1})_1$ by (C5).
- σ_{j+1} is a basic program, i.e. $\sigma_{j+1} = \alpha$ for some $\alpha \in \Sigma(A) \cap \mathcal{B}$. As $(u_j, u_{j+1}) \in [\![\alpha]\!]_{\mathcal{K}}$, there exists some $\beta \in \mathcal{B}$ such that $\mathcal{T}(u_{j+1})_2 = \beta$ and $\mathrm{At}(\alpha) = \mathrm{At}(\beta)$. Too, the above claim yields $(\mathcal{T}(u_j)_1, \mathcal{T}(u_{j+1})_1) \models \alpha$. Hence, by (L2), we get $[A_{q_{j+1}}]\chi \in \mathcal{T}(u_{j+1})_1$.

□

Continuation of Proof of Theorem 9.

Let us define some auxiliary formulas and programs. First, for every subset $X \subseteq \Lambda$, we define φ_X as

$$\varphi_X \quad = \quad \bigvee_{x \in X} p_x.$$

The formula φ_{sound} checks that the atomic propositions in the current world describe sound representations of cells of configurations of $\mathcal{M}$. More precisely, for each $0 \le i \le N-1$ exactly one of the atomic propositions $p_{c_i(0)}$ and $p_{c_i(1)}$ is true. Moreover, for exactly one element $\gamma \in \Gamma$ we have that p_γ holds. Furthermore, for at most one $q \in Q$ the atomic proposition p_q holds.

$$\begin{aligned}\varphi_{\text{sound}} \quad = \quad & \bigwedge_{0 \le i \le N-1} (c_i(0) \leftrightarrow \neg c_i(1)) \wedge \bigvee_{\gamma \in \Gamma} \left(p_\gamma \wedge \bigwedge_{\gamma' \in \Gamma : \gamma' \neq \gamma} \neg p_{\gamma'} \right) \\ & \wedge \bigwedge_{q \in Q} \left(p_q \to \bigwedge_{q' \in Q : q' \neq q} \neg p_{q'} \right)\end{aligned}$$

The formula φ_{first} (φ_{last}) checks if the current world represents the first (last) cell of some configuration:

$$\begin{aligned}\varphi_{\text{first}} &= \bigwedge_{0 \leq i \leq N-1} c_i(0)\\ \varphi_{\text{last}} &= \bigwedge_{0 \leq i \leq N-1} c_i(1)\end{aligned}$$

The formula φ_{succ} checks that the current world, say it represents a cell on position $0 \leq k \leq 2^N - 1$, does not represent the last cell (i.e. $k < 2^{N-1}$) and all successor worlds represent some cell on position $k+1$:

$$\begin{aligned}\varphi_{\text{succ}} = & \bigvee_{0 \leq i \leq N-1} \Bigg[c_i(0) \wedge [s]c_i(1) \wedge \bigwedge_{0 \leq j < i} c_j(1) \wedge [s]c_j(0)\\ & \wedge \bigwedge_{i < j \leq N-1} \Bigg(\bigwedge_{b \in \{0,1\}} c_j(b) \rightarrow [s]c_j(b) \Bigg) \Bigg]\end{aligned}$$

The formula φ_{global} guarantees that all reachable worlds describe a sound representation of cells. Moreover, it is checked that all reachable worlds, which do not represent a last cell, have some successor world that represents a cell on the successor position, and all successors of reachable worlds that represent last cells, are first cells. Finally, the formula φ_{global} prohibits that a configuration contains two cells that both contain some state symbol:

$$\begin{aligned}\varphi_{\text{global}} = & [s^*] \Bigg(\varphi_{\text{sound}} \wedge (\neg\varphi_{\text{last}} \rightarrow (\varphi_{\text{succ}} \wedge \langle s \rangle \mathtt{true})) \wedge (\varphi_{\text{last}} \rightarrow [s]\varphi_{\text{first}})\\ & \wedge \neg \left\langle (\varphi_Q \wedge \neg\varphi_{\text{last}})? \circ (s \circ \neg\varphi_{\text{first}}?)^+ \circ \varphi_Q? \right\rangle \mathtt{true} \Bigg)\end{aligned}$$

Recall that $w = w_1 \cdots w_n$ is the input. We will give a formula φ_w that guarantees that the configuration $q_0 w \Box^{2^N - n}$ is the first configuration seen.

$$\begin{aligned}\varphi_w = & \varphi_{\text{first}} \wedge p_{q_0} \wedge p_{w_1}\\ & \wedge [s] \Bigg(p_{w_2} \wedge \neg\varphi_Q\\ & \quad \wedge [s](p_{w_3} \wedge \neg p_Q\\ & \qquad \vdots\\ & \qquad \wedge [s](p_{w_n} \wedge \neg\varphi_Q) \cdots)\\ & \qquad\quad \wedge [(s \circ \neg\varphi_{\text{first}}?)^+] p_\Box \wedge \neg\varphi_Q) \Bigg)\end{aligned}$$

Let us now, for each formula ψ, define a program $\mathrm{match}(\psi)$ that relates the current cell c to the corresponding cell c' of some successor configuration (i.e. c and c' are cells at same positions) and moreover, if c and c' are not first cells, checks if the predecessor cell of c' satisfies the formula ψ. First, let us define the auxiliary program $\alpha_{-1}(\psi)$:

$$\begin{aligned}\alpha_{-1}(\psi) \quad &= \quad ((s \circ \neg\varphi_{\text{first}}?)^* \circ s \circ \varphi_{\text{first}}? \circ (s \circ \neg\varphi_{\text{first}}?)^* \circ \psi? \circ s \circ \neg\varphi_{\text{first}}?) \\ & \quad\ \cup (\varphi_{\text{first}}? \circ (s \circ \neg\varphi_{\text{first}}?)^* \circ s \circ \varphi_{\text{first}}?)\end{aligned}$$

For all $0 \leq i \leq N-1$, define auxiliary programs $\alpha_i(\psi)$:

$$\alpha_i(\psi) \quad = \quad \oplus\Big(\oplus\Big(\alpha_{i-1}(\psi), c_i(0), c_i(1)\Big), c_i(1), c_i(0)\Big)$$

Eventually, we define

$$\mathrm{match}(\psi) \quad = \quad \alpha_{N-1}(\psi).$$

Since we will enforce that each (reachable) world in a model satisfies exactly one of the atomic propositions $c_i(0)$ and $c_i(1)$ for each $0 \leq i \leq N-1$, the programs $\mathrm{match}(\psi)$ can only relate worlds that agree on exactly the same atomic propositions from $\{c_i(0), c_i(1) \mid 0 \leq i \leq N-1\}$.
Now, for every move $\mu = (q, \gamma, d) \in \text{Moves}$, where $q \in Q, \gamma \in \Gamma$, and $d \in \{\leftarrow, \rightarrow\}$, we define a program π_μ that checks if the successor configuration is in state q, scans the correct cell after moving the read/write head in direction d, and has the symbol γ at the cell that corresponds to the current one.
First, let $\mu = (q, \gamma, \rightarrow)$. As we move the read/write head of $\mathcal{M}$ to the right, we have to make sure that we do not scan the last cell of our current configuration and we define

$$\pi_\mu \quad = \quad \neg\varphi_{\text{last}}? \circ \mathrm{match}(\texttt{true}) \circ p_\gamma? \circ s \circ p_q?.$$

Analogously, since we move the read/write head to the left, we have to make sure that we do not scan the first cell of our current cnfiguration and we define

$$\pi_\mu \quad = \quad \neg\varphi_{\text{first}}? \circ \mathrm{match}(p_q) \circ p_\gamma?.$$

Let us now describe the formula φ_δ that assures that every model of φ contains a representation of a computation tree of $\mathcal{M}$.

$$\begin{aligned}\varphi_\delta \quad = \quad [s^*]\Bigg[& \bigwedge_{\substack{q \in Q_\exists, \gamma \in \Gamma: \\ \delta(q,\gamma) = (\mu_1, \mu_2)}} \Bigg((p_q \wedge p_\gamma) \rightarrow \bigvee_{i \in \{1,2\}} \langle \pi_{\mu_i} \rangle \texttt{true}\Bigg) \\ & \wedge \bigwedge_{\substack{q \in Q_\forall, \gamma \in \Gamma: \\ \delta(q,\gamma) = (\mu_1, \mu_2)}} \Bigg((p_q \wedge p_\gamma) \rightarrow \bigwedge_{i \in \{1,2\}} \langle \pi_{\mu_i} \rangle \texttt{true})\Bigg) \\ & \wedge \bigwedge_{\gamma \in \Gamma} ((p_\gamma \wedge \neg p_Q) \rightarrow [\mathrm{match}(\texttt{true})] p_\gamma)\Bigg]\end{aligned}$$

To ensure that every reachable configuration is accepting, we enforce that for all $q \in Q_{\mathrm{rej}}$, the atomic proposition p_q never occurs:

$$\varphi_{\neg\mathrm{rej}} \quad = \quad [s^*] \bigwedge_{q \in Q_{\mathrm{rej}}} \neg p_q$$

Eventually, we define

$$\varphi \quad = \quad \varphi_{\mathrm{global}} \wedge \varphi_w \wedge \varphi_\delta \wedge \varphi_{\neg\mathrm{rej}}.$$

$\square$