

Correctness and Worst-Case Optimality of Pratt-Style Decision Procedures for Modal and Hybrid Logics

Mark Kaminski¹, Thomas Schneider² and Gert Smolka¹
¹Saarland University ²University of Bremen

April 21, 2011

We extend Pratt's worst-case optimal decision procedure for PDL to a richer logic with nominals, difference modalities, and inverse actions. We prove correctness and worst-case optimality. Our correctness proof is based on syntactic models called demos. The main theorem states that a formula is satisfiable if and only if it is contained in a demo. From this theorem the correctness of the decision procedure is easily obtained. Our development is modular and we extend it stepwise from modal logic with eventualities to the full logic.

1 Introduction

Propositional dynamic logic (PDL) is an expressive extension of modal logic designed for reasoning about properties of programs and goes back to Fischer and Ladner [9]. Its satisfiability problem is EXPTIME-complete [24], and the first worst-case optimal decision procedure was given in [25]. Nominals are the basic feature of hybrid logic, which extends modal logic and goes back to Arthur Prior [26]. Nominals denote single states in models, allowing to express properties that are not expressible in standard modal logic, such as irreflexivity. The difference modality D says that a property holds in some state different from the current state, and was first described in [28]. It can be simulated using nominals and the global modality E via a satisfiability-preserving translation [11]. Nominals can be expressed using \bar{D} , the dual of D .

We consider combinations of PDL with nominals, difference modalities and converse actions, and we are interested in worst-case optimal decision proce-

dures for such combinations. Let $\text{HPDL}_{\bar{D}}$ denote the logic that combines all these features. Its computational complexity is known: the satisfiability problem is EXPTIME -complete. The lower bound follows from that for PDL by Fischer and Ladner [10]. The upper bound is due to [5, 1] via a chain of reductions that consecutively replaces D with E, removes E and converse, and ends in PDL.

The bounded model property of PDL [10]—every satisfiable formula s is satisfiable in a model of size exponential in $|s|$ —yields a straightforward guess-and-check decision procedure, whose determinization requires doubly exponential time. Pratt devised a worst-case optimal decision procedure for PDL in [25], based on Hintikka structures as a nonstandard notion of a model. These consist of Hintikka sets—consistent, downward saturated theories—and syntactic links. The search for a model is performed using tree-shaped tableaux of potentially infinite size. Using the classical filtration argument from [10] underlying the bounded model theorem (BMT), the possibly infinite tableau is filtered into a graph-shaped tableau of at most exponential size, and a straightforward procedure for searching a subgraph that represents a satisfying model is applied. In [24], Pratt describes a much leaner worst-case optimal procedure that, again, starts from all Hintikka subsets of the given formula’s closure and then deletes those that contain unsatisfied diamonds. The resulting substructure contains a satisfying model if one exists. We call this type of procedure Pratt-style and its two stages *construct* and *prune*. Pratt’s procedure is described in [16, 22, 17, 3], where [3] uses a stricter notion of Hintikka sets and excludes tests.

A practical problem with Pratt-style procedures is that the initial *construct* stage is “best-case exponential”, although certainly not every Hintikka set plays a role in a satisfying model. This problem is reduced in decision procedures based on (non-branching) tableaux, such as Pratt’s procedure in [25]. They make *construct* more goal-directed by restricting the creation of new nodes—representatives of Hintikka sets—to those that reduce formulas in nodes already present. Such tableau-based procedures exist for different modal-like logics and are often optimized further by interleaving *construct* and *prune* [13, 14].

Decision procedures based on branching tableau systems [27, 18, 6, 4, 21] enjoy wide regard in automated reasoning with modal and description logics. They typically run in worst-case non-deterministic doubly exponential time, but highly optimized systems work well in practice [15, 30]. However, there are exponential-time algorithms based on branching tableaux for description logics [7, 12].

Automata-theoretic decision procedures exploit some form of tree-model property of the logic in question, transfer a given formula into an automaton of typically exponential size, and thus reduce satisfiability to the emptiness problem of the automata model corresponding to the logic. This approach is applied to expressive modal logics extending PDL [31, 29]. However, in general, the

complexity is “best-case exponential” again: the whole automaton needs to be constructed. Whenever new features are added to the logic, the automata model needs to be adjusted and, often, the complexity proof for the emptiness problem redone.

This paper presents a modular approach to obtaining lean proofs of the BMT and the correctness of worst-case optimal Pratt-style decision procedures for the above mentioned extensions of PDL. These decision procedures will be able to handle hybrid operators in an additional deterministic guess stage. We use the notion of a demo—a syntactic representation of a satisfying model in terms of Hintikka sets. With this notion, we tailor the proofs of the BMT for said logics to the expressive features involved. We will analyze the conceptual, technical and computational costs required for incorporating each of those features, as well as their combinability.

The strengths of the modular approach are the following.

- We refactor the standard proofs leading to the the BMT such that standard induction over term lengths suffices.
- The explicit use of demos makes the BMT proofs transparent and reusable for the correctness of the decision procedure.
- The addition of the above named expressive features is modular: different features can be added independently by combining the techniques needed for every single feature.
- To our knowledge, this is the first explicit and simple worst-case optimal decision procedure for a logic that combines PDL and hybrid operators.

The paper is organized as follows. We will introduce hybrid PDL, introduce demos for test-free PDL and discuss their relevant properties, present the decision procedure, discuss extensions of the language separately, and relate this approach to those in the literature.

2 Preliminaries: hybrid PDL

Let PRED and ACT be countably infinite sets, whose elements are called **predicates** and **actions**, respectively. Let $\text{NOM} \subseteq \text{PRED}$ be the set of all **nominals**. We assume formulas to be in negation normal form (NNF), i.e., negation is allowed to occur only directly in front of predicates. We also assume programs to be in converse normal form (CNF), i.e., the converse operator is allowed to occur only directly after actions. Formulas s and programs α of $\text{HPDL}_{\bar{\text{D}}}$ are defined by

mutual recursion as follows, where $p \in \text{PRED}$ and $a \in \text{ACT}$.

$$\begin{aligned} s ::= & p \mid s \wedge s \mid \langle \alpha \rangle s \mid \mathbb{D}s \mid \\ & \neg p \mid s \vee s \mid [\alpha]s \mid \overline{\mathbb{D}}s \\ \alpha ::= & a \mid a^- \mid \alpha\beta \mid \alpha+\beta \mid \alpha^* \mid s \end{aligned}$$

We denote predicates by p, q, \dots , nominals by x, y, \dots , formulas by s, t, \dots , actions by a, b, \dots and programs by α, β, \dots . The operator \mathbb{D} is called the difference modality, and $\overline{\mathbb{D}}$ is its dual. If we want to denote the fragment of $\text{HPDL}_{\overline{\mathbb{D}}}$ without converse, nominals, and/or difference modalities, we leave out the superscript “-”, the leading H and/or the subscript \mathbb{D} .

Choosing to adapt NNF and CNF is not crucial for our approach to work. It merely simplifies the presentation, even though it increases some basic definitions by dual cases. It is no computational obstacle either: any formula can be transformed into an equivalent formula in NNF and CNF in linear time.

In order to capture tests as programs, and only for this purpose, we use Tait negation: $\sim s$ denotes the NNF of $\neg s$, with the obvious consequence $\sim \sim s = s$. We further use the notation $|s|$ to denote the size of a formula, which is defined as usual, with the only exception being $|\neg p| = |p|$. This ensures that $|\sim s| = |s|$.

We recall the standard operations on binary relations R, S over a set X .

$$\begin{aligned} R^- &= \{(x, y) \mid (y, x) \in R\} \\ R \circ S &= \{(x, z) \mid \exists y \in X : (x, y) \in R \text{ and } (y, z) \in S\} \\ R^0 &= \{(x, x) \mid x \in X\} \\ R^n &= R \circ R^{n-1}, \quad \text{for } n \geq 1 \\ R^* &= \bigcup_{n \geq 0} R^n \end{aligned}$$

As usual, the semantics of $\text{HPDL}_{\overline{\mathbb{D}}}$ is defined in terms of Kripke models. A **model** \mathfrak{M} consists of

- a nonempty set $|\mathfrak{M}|$ of states,
- a **transition relation** $\xrightarrow{a}_{\mathfrak{M}} \subseteq |\mathfrak{M}| \times |\mathfrak{M}|$ for every $a \in \text{ACT}$,
- a set $\mathfrak{M}p \subseteq |\mathfrak{M}|$ for every $p \in \text{PRED}$, where $|\mathfrak{M}x| = 1$ for every $x \in \text{NOM}$.

The transition relations for complex programs and the **satisfaction relation** between models, their states and formulas, written $\mathfrak{M}, w \models s$, are defined via mu-

tual induction.

$$\begin{aligned}
\frac{a^-}{\rightarrow_{\mathcal{M}}} &= \frac{a}{\rightarrow_{\mathcal{M}}}^- \\
\frac{\alpha\beta}{\rightarrow_{\mathcal{M}}} &= \frac{\alpha}{\rightarrow_{\mathcal{M}}} \circ \frac{\beta}{\rightarrow_{\mathcal{M}}} \\
\frac{\alpha+\beta}{\rightarrow_{\mathcal{M}}} &= \frac{\alpha}{\rightarrow_{\mathcal{M}}} \cup \frac{\beta}{\rightarrow_{\mathcal{M}}} \\
\frac{\alpha^*}{\rightarrow_{\mathcal{M}}} &= \frac{\alpha}{\rightarrow_{\mathcal{M}}}^* \\
\frac{s}{\rightarrow_{\mathcal{M}}} &= \{(w, w) \mid \mathcal{M}, w \models s\}
\end{aligned}$$

$$\begin{aligned}
\mathcal{M}, w \models p &\iff w \in \mathcal{M}p \quad \text{for } p \in \text{PRED} \\
\mathcal{M}, w \models \neg p &\iff w \notin \mathcal{M}p \quad \text{for } p \in \text{PRED} \\
\mathcal{M}, w \models s \wedge t &\iff \mathcal{M}, w \models s \text{ and } \mathcal{M}, w \models t \\
\mathcal{M}, w \models \langle \alpha \rangle s &\iff \mathcal{M}, v \models s \text{ for some } v \in |\mathcal{M}| \text{ with } w \xrightarrow{\alpha}_{\mathcal{M}} v \\
\mathcal{M}, w \models Ds &\iff \mathcal{M}, v \models s \text{ for some } v \neq w
\end{aligned}$$

The satisfaction relation for the remaining operators can be obtained from the equivalences $s \vee t \equiv \neg(\neg s \wedge \neg t)$, $[\alpha]s \equiv \neg\langle \alpha \rangle \neg s$, and $\bar{D}s \equiv \neg D\neg s$.

We extend the notion of satisfaction to sets A of formulas in the obvious way: $\mathcal{M}, w \models A$ if $\mathcal{M}, w \models s$ for all $s \in A$.

A **literal** is a formula of the form p , $\neg p$, $\langle a \rangle s$, $[a]s$, $\langle a^- \rangle s$, $[a^-]s$, Ds or $\bar{D}s$, where $p \in \text{PRED}$, $a \in \text{ACT}$, and s is an arbitrary formula.

A **Hintikka set is a partial description of a possible state**. It contains formulas satisfied by that state. A system of Hintikka sets then represents a satisfying model, and our goal is to show that every satisfiable formula is contained in a Hintikka set that is part of a finite such system. More precisely, a **Hintikka set** is a nonempty set H that satisfies the following properties.

- For every $p \in \text{PRED}$: $\{p, \neg p\} \not\subseteq H$.
- $s \wedge t \in H \implies s \in H$ and $t \in H$
- $s \vee t \in H \implies s \in H$ or $t \in H$
- $\langle \alpha\beta \rangle s \in H \implies \langle \alpha \rangle \langle \beta \rangle s \in H$
- $[\alpha\beta]s \in H \implies [\alpha][\beta]s \in H$
- $\langle \alpha+\beta \rangle s \in H \implies \langle \alpha \rangle s \in H$ or $\langle \beta \rangle s \in H$
- $[\alpha+\beta]s \in H \implies [\alpha]s \in H$ and $[\beta]s \in H$
- $\langle \alpha^* \rangle s \in H \implies \langle \alpha \rangle \langle \alpha^* \rangle s \in H$ or $s \in H$
- $[\alpha^*]s \in H \implies [\alpha][\alpha^*]s \in H$ and $s \in H$
- $\langle t \rangle s \in H \implies t \in H$ and $s \in H$
- $[t]s \in H \implies \sim t \in H$ or $s \in H$

A **Hintikka system** S is a finite, nonempty set of Hintikka sets. We say that a formula s is **contained in S** if it is contained in some $H \in S$.

In order to restrict the choice of possible elements of a Hintikka set, we assume a finite, nonempty **formula universe** \mathcal{F} , which is modelled on the Fischer-Ladner closure [10] of a given formula s . \mathcal{F} consists of formulas in NNF and satisfies the following closure properties.

- $s \in \mathcal{F}$ and t is a subformula of $s \Rightarrow t \in \mathcal{F}$
- $\langle \alpha \beta \rangle s \in \mathcal{F} \Rightarrow \langle \alpha \rangle \langle \beta \rangle s \in \mathcal{F}$
- $[\alpha \beta] s \in \mathcal{F} \Rightarrow [\alpha][\beta] s \in \mathcal{F}$
- $\langle \alpha + \beta \rangle s \in \mathcal{F} \Rightarrow \langle \alpha \rangle s, \langle \beta \rangle s \in \mathcal{F}$
- $[\alpha + \beta] s \in \mathcal{F} \Rightarrow [\alpha] s, [\beta] s \in \mathcal{F}$
- $\langle \alpha^* \rangle s \in \mathcal{F} \Rightarrow \langle \alpha \rangle \langle \alpha^* \rangle s \in \mathcal{F}$
- $[\alpha^*] s \in \mathcal{F} \Rightarrow [\alpha][\alpha^*] s \in \mathcal{F}$
- $[t] s \in \mathcal{F} \Rightarrow \sim t \in \mathcal{F}$

This is a slight variation of the definitions in the literature [9, 22, 17]. Still, we can use the following original result.

Lemma 2.1 ([9]) For every formula s , one can compute a finite formula universe \mathcal{F} such that $s \in \mathcal{F}$ and the cardinality of \mathcal{F} is linear in the size of s .

From now on, all formulas, Hintikka sets and systems range over a given \mathcal{F} .

3 Demos as a syntactic representation of models

We aim at the following criterion for syntactically **demonstrating** that a given formula s is satisfiable: s is satisfiable if and only if s occurs in a Hintikka system S that sufficiently describes a model. We call such a system a demo. A maximal demo corresponds to the result of any of the elimination procedures for Hintikka systems described in [25, 17, 3]. The notion of a demo derives from that of an evident subset of a Pratt-style graph tableau in [20]. Making the demo notion explicit will allow for factoring the bounded model theorem into lemmas that use simpler inductions. The main lemmas, demo existence and satisfaction lemmas will almost immediately imply correctness of the decision procedure.

In this section, we introduce the notion of a demo and study it in depth. In order to keep the presentation simple, we begin with test-free PDL and will add nominals, difference modalities, tests and converse separately in Sections 5–8. This will allow us to isolate the conceptual, technical, and computational cost of adding those features.

Definition 3.1 Let S be a Hintikka system. The **transition relation** $\xrightarrow{\alpha}_S \subseteq S \times S$ is defined as follows.

$$\begin{aligned}\xrightarrow{a}_S &= \{(H, H') \mid \text{for all } s : ([a]s \in H \Rightarrow s \in H')\} \\ \xrightarrow{\alpha\beta}_S &= \xrightarrow{\alpha}_S \circ \xrightarrow{\beta}_S \\ \xrightarrow{\alpha+\beta}_S &= \xrightarrow{\alpha}_S \cup \xrightarrow{\beta}_S \\ \xrightarrow{\alpha^*}_S &= \xrightarrow{\alpha}_S^*\end{aligned}$$

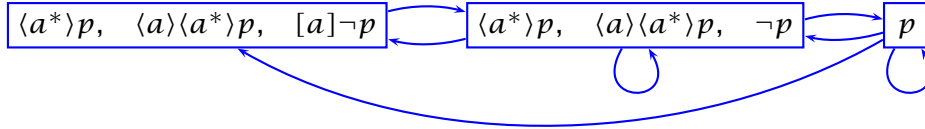
Proposition 3.2 Let $S \subseteq S'$ be Hintikka systems. Then $\xrightarrow{\alpha}_S \subseteq \xrightarrow{\alpha}_{S'}$.

Definition 3.3 A Hintikka system \mathcal{D} is a **demo** if the following condition is satisfied.

(D \diamond) If $\langle \alpha \rangle s \in H \in \mathcal{D}$, then there is some $H' \in \mathcal{D}$ with $H \xrightarrow{\alpha}_{\mathcal{D}} H'$ and $s \in H'$.

It suffices to require (D \diamond) only for programs α that are actions or iterations β^* . The remaining cases would then follow via the definition of Hintikka sets. For clarity of the presentation, however, we do not make this restriction.

Example 3.4 The figure below shows a demo that consists of three Hintikka sets: $\{\langle a^* \rangle p, \langle a \rangle \langle a^* \rangle p, [a] \neg p\}$, $\{\langle a^* \rangle p, \langle a \rangle \langle a^* \rangle p, \neg p\}$ and $\{p\}$. Sets related by \xrightarrow{a} are connected with an arrow. Arrows for $\xrightarrow{a^*}$ are implicit. \square



The following fact is obvious from the formulation of Demo Condition (D \diamond): demos are closed under union, and therefore there is a unique maximal demo for \mathcal{F} . This will be important for the correctness of the decision procedure.

In order to establish that demos represent exactly models modulo \mathcal{F} , we show that (a) every model induces a demo in the natural way, and (b) every demo has a model that satisfies all of the demo's members. We start with (a).

Given a model \mathfrak{M} and a state $w \in |\mathfrak{M}|$, let $H_{\mathfrak{M},w} = \{s \in \mathcal{F} \mid \mathfrak{M}, w \models s\}$ be the Hintikka set induced by w in \mathfrak{M} . The following proposition is obvious.

Proposition 3.5 $H_{\mathfrak{M},w}$ is a Hintikka set.

From now on, we write H_w instead of $H_{\mathfrak{M},w}$ if no confusion can arise. We now consider the system of all Hintikka sets induced by \mathfrak{M} , i.e., $S_{\mathfrak{M}} = \{H_w \mid w \in |\mathfrak{M}|\}$. In order to establish that $S_{\mathfrak{M}}$ is a demo, we need the following lemma.

Lemma 3.6 Let \mathfrak{M} be a model and $v, w \in |\mathfrak{M}|$. If $v \xrightarrow{\alpha}_{\mathfrak{M}} w$, then $H_v \xrightarrow{\alpha}_{S_{\mathfrak{M}}} H_w$.

Proof We proceed by induction on the size of α .

$\alpha = a$. Since $v \xrightarrow{a}_{\mathfrak{M}} w$, we have that $\mathfrak{M}, w \models s$ for all $[a]s$ with $\mathfrak{M}, v \models [a]s$. This implies that $s \in H_w$ whenever $[a]s \in H_v$. Hence, $H_v \xrightarrow{a}_{S_{\mathfrak{M}}} H_w$.

$\alpha = \beta\gamma$. If $v \xrightarrow{\beta\gamma}_{\mathfrak{M}} w$, there is some $u \in |\mathfrak{M}|$ with $v \xrightarrow{\beta}_{\mathfrak{M}} u \xrightarrow{\gamma}_{\mathfrak{M}} w$. Applying the induction hypothesis to β and γ yields $H_v \xrightarrow{\beta}_{S_{\mathfrak{M}}} H_u \xrightarrow{\gamma}_{S_{\mathfrak{M}}} H_w$, i.e., $H_v \xrightarrow{\beta\gamma}_{S_{\mathfrak{M}}} H_w$.

$\alpha = \beta + \gamma$. If $v \xrightarrow{\beta + \gamma}_{\mathfrak{M}} w$, then $v \xrightarrow{\beta}_{\mathfrak{M}} w$ or $v \xrightarrow{\gamma}_{\mathfrak{M}} w$. Applying the induction hypothesis to β and γ yields $H_v \xrightarrow{\beta}_{S_{\mathfrak{M}}} H_w$ or $H_v \xrightarrow{\gamma}_{S_{\mathfrak{M}}} H_w$, i.e., $H_v \xrightarrow{\beta + \gamma}_{S_{\mathfrak{M}}} H_w$.

$\alpha = \beta^*$. If $v \xrightarrow{\beta^*}_{\mathfrak{M}} w$, then $v \xrightarrow{\beta}_{\mathfrak{M}}^n w$ for some $n \geq 0$. We proceed by induction on n . If $n = 0$, then $v = w$ and therefore $H_v = H_w$, i.e., $H_v \xrightarrow{\beta^*}_{S_{\mathfrak{M}}} H_w$. If $n > 0$, there is some $u \in |\mathfrak{M}|$ with $H_v \xrightarrow{\beta}_{S_{\mathfrak{M}}} H_u \xrightarrow{\beta}_{S_{\mathfrak{M}}}^{n-1} H_w$. From both induction hypotheses, we obtain $H_v \xrightarrow{\beta^*}_{S_{\mathfrak{M}}} H_w$. ■

Lemma 3.7 $S_{\mathfrak{M}}$ is a demo.

Proof Let $\langle \alpha \rangle s \in H_w \in S_{\mathfrak{M}}$. Then $\mathfrak{M}, w \models \langle \alpha \rangle s$, that is, there is some $v \in |\mathfrak{M}|$ such that $w \xrightarrow{\alpha}_{\mathfrak{M}} v$ and $\mathfrak{M}, v \models s$. Due to Lemma 3.6 and the definition of H_v , we obtain $H_w \xrightarrow{\alpha}_{S_{\mathfrak{M}}} H_v$ and $s \in H_v$. ■

Lemma 3.8 (Demo existence) For every satisfiable formula $s \in \mathcal{F}$, there is a demo \mathcal{D} over \mathcal{F} that contains s .

Proof Let $\mathfrak{M}, w \models s$. Take $\mathcal{D} = S_{\mathfrak{M}}$, which contains H_w with $s \in H_w$. ■

As for the direction (b) above, we start by making some general statements about Hintikka systems. The first such statement is that S , together with the transition relations \xrightarrow{a}_S , induces a model \mathfrak{M}_S as follows.

Definition 3.9 Let S be a Hintikka system. \mathfrak{M}_S is the model defined as follows.

$$\begin{aligned} |\mathfrak{M}_S| &= S \\ \xrightarrow{a}_{\mathfrak{M}_S} &= \xrightarrow{a}_S \\ \mathfrak{M}_S p &= \{H \in S \mid p \in H\} \end{aligned}$$

In general, \mathfrak{M}_S does not need to satisfy the Hintikka sets in S . However, under additional conditions categorized under the notion of a demo, there is a direct correspondence between Hintikka systems and models. .

Lemma 3.10 Let S be a Hintikka system and α a program. Then $\xrightarrow{\alpha}_S \subseteq \xrightarrow{\alpha}_{\mathfrak{W}_S}$.

Proof We proceed by induction on α .

$\alpha = a$. The claim holds due to the definition of \mathfrak{W}_S .

$\alpha = \beta\gamma$. Let $H \xrightarrow{\beta\gamma}_S H'$. Then there is some $H'' \in S$ with $H \xrightarrow{\beta}_S H'' \xrightarrow{\gamma}_S H'$. We apply the induction hypothesis: $H \xrightarrow{\beta}_{\mathfrak{W}_S} H'' \xrightarrow{\gamma}_{\mathfrak{W}_S} H'$, i.e., $H \xrightarrow{\beta\gamma}_{\mathfrak{W}_S} H'$.

$\alpha = \beta + \gamma$. Let $H \xrightarrow{\beta + \gamma}_S H'$. Then, $H \xrightarrow{\beta}_S H'$ or $H \xrightarrow{\gamma}_S H'$. With the induction hypothesis, we obtain $H \xrightarrow{\beta}_{\mathfrak{W}_S} H'$ or $H \xrightarrow{\gamma}_{\mathfrak{W}_S} H'$, i.e., $H \xrightarrow{\beta + \gamma}_{\mathfrak{W}_S} H'$.

$\alpha = \beta^*$. Let $H \xrightarrow{\beta^*}_S H'$, i.e., $H \xrightarrow{\beta}_S^n H'$ for some $n \geq 0$. We proceed by induction on n . The case $n = 0$ is trivial. If $n > 0$, we have $H \xrightarrow{\beta}_S H'' \xrightarrow{\beta}_{S}^{n-1} H'$ for some $H'' \in S$. From both induction hypotheses, we obtain $H \xrightarrow{\beta^*}_{\mathfrak{W}_S} H'$. ■

Lemma 3.11 Let S be a Hintikka system with $H, H' \in S$ and $[\alpha]s \in H \xrightarrow{\alpha}_{\mathfrak{W}_S} H'$. Then $s \in H'$.

Proof We proceed by induction on the size of α .

$\alpha = a$. We obtain $s \in H'$ directly from the definition of $\xrightarrow{a}_{\mathfrak{W}_S}$.

$\alpha = \beta\gamma$. Due to the definition of $\xrightarrow{\beta\gamma}_{\mathfrak{W}_S}$, there is a $H'' \in S$ with $H \xrightarrow{\beta}_{\mathfrak{W}_S} H'' \xrightarrow{\gamma}_{\mathfrak{W}_S} H'$. Since H is a Hintikka set, we obtain $[\beta][\gamma]s \in H$. Applying the induction hypothesis to β and then γ yields $[y]s \in H''$ and $s \in H'$.

$\alpha = \beta + \gamma$. Since H is a Hintikka set, we obtain $[\beta]s \in H$ and $[\gamma]s \in H$. From $H \xrightarrow{\beta + \gamma}_{\mathfrak{W}_S} H'$, we obtain $H \xrightarrow{\beta}_{\mathfrak{W}_S} H'$ or $H \xrightarrow{\gamma}_{\mathfrak{W}_S} H'$. The induction hypothesis applied to either β or γ yields $s \in H'$.

$\alpha = \beta^*$. Since H is a Hintikka set, we obtain (i) $s \in H$ and (ii) $[\beta][\beta^*]s \in H$. From $H \xrightarrow{\beta^*}_{\mathfrak{W}_S} H'$ we conclude $H \xrightarrow{\beta}_{\mathfrak{W}_S}^n H'$ for some $n \geq 0$. We proceed by induction on n . The case $n = 0$ is trivial given (i). If $n > 0$, there is some $H'' \in S$ with $H \xrightarrow{\beta}_{\mathfrak{W}_S} H'' \xrightarrow{\beta}_{\mathfrak{W}_S}^{n-1} H'$. Using (ii), we apply the outer induction hypothesis to the first arrow and obtain $[\beta^*]s \in H''$. We then apply the inner induction hypothesis to the second arrow and obtain $s \in H'$. ■

Lemma 3.12 (Demo satisfaction) If \mathcal{D} is a demo then, for all $H \in \mathcal{D} : \mathfrak{W}_{\mathcal{D}}, H \models H$.

Proof We take $s \in H \in \mathcal{D}$ and prove $\mathfrak{M}_{\mathcal{D}}, H \models s$ by induction on the size of s . The case $s = p$ follows from the definition of $\mathfrak{M}_{\mathcal{D}}p$. The Boolean cases are straightforward. The cases of diamond and box formulas remain.

$s = \langle \alpha \rangle t$. If $\langle \alpha \rangle t \in H \in \mathcal{D}$, then Demo Condition ($D\Diamond$) requires the existence of an $H' \in \mathcal{D}$ with $H \xrightarrow{\alpha}_{\mathcal{D}} H'$ and $t \in H'$. We can now apply Lemma 3.10, whose additional assumption is satisfied due to the induction hypothesis, to conclude $H \xrightarrow{\alpha}_{\mathfrak{M}_{\mathcal{D}}} H'$. The induction hypothesis also yields $\mathfrak{M}_{\mathcal{D}}, H' \models t$. Therefore, $\mathfrak{M}_{\mathcal{D}}, H \models \langle \alpha \rangle t$.

$s = [\alpha]t$. If $[\alpha]t \in H \in \mathcal{D}$, then we can apply Lemma 3.11, whose additional assumption is satisfied due to the induction hypothesis, to conclude that $t \in H'$ for every $H' \in \mathcal{D}$ with $H \xrightarrow{\alpha}_{\mathfrak{M}_{\mathcal{D}}} H'$. The induction hypothesis further yields $\mathfrak{M}_{\mathcal{D}}, H' \models t$. Therefore, $\mathfrak{M}_{\mathcal{D}}, H \models [\alpha]t$. ■

The following central insight about demos follows directly from Lemmas 3.8, 3.12.

Theorem 3.13 A formula s is satisfiable if and only if s is contained in a demo.

Lemmas 3.8 and 3.12 also imply the Bounded Model Theorem for test-free PDL, which has been established in [10] for PDL: every satisfiable formula s is satisfiable in a finite model of size exponential in $|s|$.

4 The decision procedure

We use Pratt's approach of constructing the set of all Hintikka sets and pruning it to the greatest demo. The correctness of this procedure will immediately follow from the fact that pruning respects demos: every single pruning action does not remove any Hintikka set that is part of a demo contained in the system before the pruning action. This argument exploits the existence of a maximal demo, which is not guaranteed in the presence of nominals or \bar{D} . We will deal with that problem in Sections 5-6.

We define a relation between Hintikka systems that represents a single pruning action: let $S \xrightarrow{p} S'$ if S' can be obtained from S by deleting some $H \in S$ that violates the Demo Condition ($D\Diamond$), i.e., for some $\langle \alpha \rangle s \in H \in S$, there is no $H' \in S$ such that $H \xrightarrow{\alpha}_S H'$ and $s \in H'$. We further define $S \xrightarrow{p^*} S'$ to hold if $S \xrightarrow{p} S'$ and $S' \xrightarrow{p} S''$ for no S'' .

The following proposition is immediate because (1) every pruning action removes only Hintikka sets that violate the demo condition, and (2) when no more pruning can be done, all Hintikka sets in S' satisfy the demo condition.

Proposition 4.1

1. If $S \xrightarrow{p} S'$ and \mathcal{D} is a demo with $\mathcal{D} \subseteq S$, then $\mathcal{D} \subseteq S'$.
2. If $S \xrightarrow{p} S' \neq \emptyset$, then S' is a demo.

Theorem 4.2 If $S \xrightarrow{p} S'$ and S contains a demo, then S' is the greatest demo contained in S .

Proof Due to Prop. 4.1 1, S' contains all demos contained in S , including the greatest such demo. Due to 2, S' is a demo itself, hence it is contained in the greatest demo contained in S . Both inclusions together yield the equality. ■

The following method decides satisfiability of a given formula s by pruning the system of all Hintikka sets and checking whether s is contained in the resulting demo.

Decision method for PDL-satisfiability

Input: formula s

1. Compute the formula universe \mathcal{F} for s .
 2. $\mathcal{H} = \{H \mid H \text{ is a Hintikka set with } H \subseteq \mathcal{F}\}$
 3. Compute \mathcal{D} with $\mathcal{H} \xrightarrow{p} \mathcal{D}$.
 4. s is satisfiable iff $s \in H$ for some $H \in \mathcal{D}$.
-

The above decision method is a notational variant of Pratt's [24] decision procedure. By making the notion of a demo explicit. We can conclude correctness directly from Theorem 4.2, while the correctness proofs in [22, 17] use complex inductive arguments quite similar to the proofs of the filtration theorem. Steps 1 and 2 correspond to Stage construct, and Step 3 to prune. The transition relations $\xrightarrow{\alpha}_s$ are not computed upfront. Instead, whenever the decision procedure needs to decide for a pair of Hintikka sets whether they are in some $\xrightarrow{\alpha}_s$, it does so via the inductive definition of $\xrightarrow{\alpha}_s$, in time polynomial in $|\alpha|$ multiplied by the sizes of the Hintikka sets.

We now convince ourselves that the above method is worst-case optimal, i.e., that it runs in time exponential in $|s|$, looking at each step in turn.

1. Due to Lemma 2.1, the cardinality of \mathcal{F} is linear in $|s|$. Furthermore, if \mathcal{F} is taken to be the smallest formula universe that contains s , then \mathcal{F} can be computed in time polynomial in $|s|$ by following the closure properties.
2. In order to compute \mathcal{H} , one can create all exponentially many subsets of \mathcal{F} and remove those that are not Hintikka sets. Checking the Hintikka set properties of any such H requires time at most polynomial in the cardinality

of H , which in turn is linear in $|s|$. In preparation for the following step, all transition relations $\xrightarrow{\alpha}_s$ over \mathcal{H} can be precomputed and reused for every S during the pruning phase. For every α , determining $\xrightarrow{\alpha}_s$ takes time quadratic in the cardinality of \mathcal{H} .

3. Since the number of Hintikka sets decreases with every single pruning action, there can be at most exponentially many pruning actions. Each of them can be performed in time exponential in $|s|$: traverse through all Hintikka sets H in the remaining system S and all formulas $\langle \alpha \rangle s \in H$, and check whether some $H' \in S$ exists with $H \xrightarrow{\alpha}_s H'$.
4. Traversing through all Hintikka sets left and through their contents to search for s is clearly in exponential time as well.

For practical purposes, creating all Hintikka sets in the first place is highly inefficient in terms of both time and space. We have discussed possible optimizations in Section 1.

5 Nominals

Extending the demo notion with nominals is rather straightforward. We need to introduce the notion of nominal coherence: a Hintikka system \mathcal{H} is **nominally coherent** if every nominal $x \in \mathcal{F}$ occurs in exactly one $H \in S$. Now the definition of a demo (Def. 3.3) and of \mathfrak{M}_S , as well as the assumptions of Lemma 3.10 have to be extended by the requirement (DN) that \mathcal{D} and S be nominally coherent.¹

Demos for PDL with nominals are no longer closed under union, but this will not affect the proofs of Section 3. We only need to extend the proof of Lemma 3.7 by saying that $S_{\mathfrak{M}}$ satisfies (DN): if $x \in \mathcal{F}$, then x denotes a unique state $w_x \in |\mathfrak{M}|$. Therefore, x is contained in the unique induced Hintikka set H_{w_x} .

So far, the addition of nominals has been at no extra conceptual or technical cost. When it comes to pruning, however, we can no longer remove arbitrary Hintikka sets that “violate the demo conditions”. For example, if the same nominal x is contained in two Hintikka sets H, H' of the system S and we remove H , then further pruning actions carried out to restore (D \diamond) could lead to H' being deleted as well. More generally, since there may not need to be a unique maximal demo, Theorem 4.2 does not hold. However, it can be reestablished using an additional assumption here and in the preceding proposition.

Proposition 5.1 Let S be nominally coherent. Then the following hold.

¹ While it would suffice for the proofs in this section to require that every nominal $x \in \mathcal{F}$ occurs in **at most** one $H \in S$, this would not be enough in the presence of difference modalities. Otherwise, $\{\{x, \neg x, Dx\}\}$ would be a nominally coherent demo although its only member is unsatisfiable.

1. If S contains a demo, then it contains a unique maximal demo.
2. If $S \xrightarrow{p} S'$ and $\mathcal{D} \subseteq S$ is a demo, then $\mathcal{D} \subseteq S'$ and S' is nominally coherent.
3. If $S \xrightarrow{p} S'$ and S' is nominally coherent, then S is a demo.

Proof We call a Hintikka set that contains a nominal a **nominal set**.

1. Let $\mathcal{D}_1, \mathcal{D}_2$ be two demos contained in S . Then, due to $S, \mathcal{D}_1, \mathcal{D}_2$ being nominally coherent, both \mathcal{D}_i contain all nominal sets in S . Therefore and because $(D\Diamond)$ is robust under union, we have that $\mathcal{D}_1 \cup \mathcal{D}_2 \subseteq S$.
2. Every pruning action removes only Hintikka sets that violate $(D\Diamond)$; hence no Hintikka set from \mathcal{D} is removed. Because S is nominally coherent and \mathcal{D} contains all nominal sets in S , S' is nominally coherent too.
3. Since S' is nominally coherent, it cannot be empty. Because no more pruning can be done, all Hintikka sets in S' satisfy $(D\Diamond)$ as well. ■

Theorem 5.2 If $S \xrightarrow{p} S'$ with S and S' being nominally coherent, then S' is the unique maximal demo contained in S .

Proof The unique maximal demo contained in S exists because of Proposition 5.1 1; we call it \mathcal{D} . Due to 3, S' is a demo contained in S and therefore contained in \mathcal{D} . Due to 2, S' contains all demos contained in S , including \mathcal{D} . Both inclusions together yield the equality. ■

The decision method from Section 4 can now be extended by inserting a `guess` stage between Steps 2 and 3 that guesses a maximal nominally coherent subsystem of the system \mathcal{H} of all Hintikka sets, which contains one maximal demo. As explained above, the transition relations $\xrightarrow{\alpha}_S$ do not need to be computed upfront. The nondeterministic procedure is given below.

Decision method for HPDL-satisfiability

Input: formula s

1. Compute the formula universe \mathcal{F} for s .
 2. $\mathcal{H} = \{H \mid H \text{ is a Hintikka set with } H \subseteq \mathcal{F}\}$
 3. Guess a maximal nominally coherent subset \mathcal{H}' of \mathcal{H} .
 4. Compute \mathcal{D} with $\mathcal{H}' \xrightarrow{p} \mathcal{D}$.
 5. Return “satisfiable” iff \mathcal{D} is nominally coherent and $s \in H$ for some $H \in \mathcal{D}$.
-

We now show that this method has a determinization that runs in exponential time. Let x_1, \dots, x_n be the linearly many nominals in \mathcal{F} . For x_1 , Step 3 can guess

a Hintikka set in \mathcal{H} that contains x_1 and remove all other Hintikka sets that contain x_1 . This action can be iterated for all other x_i . Should no Hintikka set be left that contains x_i , then Step 3 rejects straightaway. Otherwise the reduced Hintikka system after the n -th iteration is a maximal nominally coherent subset of \mathcal{H} . Steps 4 and 5 are then applied deterministically to that subset.

We consider the set of computation paths of this nondeterministic algorithm. Every path contains n guesses of an element from a subset of \mathcal{H} —from a set exponential in $|s|$. Each such exponential guess can be implemented as a sequence of polynomially many binary guesses, inducing a binary tree of polynomial depth. Since n is linear in $|s|$, the binary tree induced by the sequence of all n exponential guesses is still of polynomial depth. In every leaf of this tree, a deterministic exponential time computation takes place. Hence, the tree has only exponentially many nodes and can be searched deterministically in exponential time.

To summarize, adding nominals requires no significant computational costs as long as only worst-case complexity is concerned. Our decision method remains correct if Step 2 is replaced by the creation of a closed tableau completed via rules extending those in [20]. However, both versions of our method are impractical because `prune` is repeated a number of times that is linear in the size of \mathcal{H} or the completed tableau. It would be more practical to interleave `guess` with `construct`, but it is currently not clear how to realize this.

6 Difference modalities

To deal with the difference operators, several changes to the conceptual, technical and computational part are necessary.

Since Ds and $\bar{D}s$ do not say anything about the current state, it is not the notion of a Hintikka set that needs extending, but the notion of a demo. We add the following conditions to Def. 3.3.

(DD) If $Ds \in H \in \mathcal{D}$, then there is some $H' \in \mathcal{D}$ such that $H' \neq H$ and $s \in H'$.

($\bar{D}\bar{D}$) If $\bar{D}s \in H \in \mathcal{D}$, then, for all $H' \in \mathcal{D}$ such that $H' \neq H$, we have $s \in H'$.

Because of Condition ($\bar{D}\bar{D}$) alone, demos for PDL extended by \bar{D} are not closed under union. This is not surprising because \bar{D} can be used to express that a given predicate behaves as a nominal: $p \wedge \bar{D}\neg p$. We will therefore have to adapt the decision procedure to the presence of \bar{D} at the end of this section. In contrast, adding only D to PDL does not make closure under unions invalid.

However, the proof of Lemma 3.7— S_{\exists} is a demo—does not go through for the D case without further assumptions. Consider, for example, $\mathcal{F} = \{p, Dp\}$ and the

model \mathfrak{M} with $|\mathfrak{M}| = \{v, w\}$ and $\mathfrak{M}p = \{v, w\}$. Then $\mathfrak{M}, v \models Dp$ and $Dp \in H_v$. Since $H_v = H_w$, the system $S_{\mathfrak{M}}$ consists of only H_v and (DD) is violated.

To solve this problem, we assume an injective function that assigns to every literal $Ds \in \mathcal{F}$ a nominal $x_{Ds} \in \mathcal{F}$ that is isolated, i.e., which occurs in no other formula in \mathcal{F} . Intuitively, x_{Ds} is supposed to denote a state that satisfies s , provided that such a state exists. If it does, all states that are different from the one denoted by x_{Ds} satisfy Ds . As we will see below, this implies that, whenever a state w satisfies Ds , then there is a state v that satisfies s with $H_v \neq H_w$.

It remains to ensure that the x_{Ds} denote the correct states in \mathfrak{M} . We call a model \mathfrak{M} **nice** for \mathcal{F} if, for all $Ds \in \mathcal{F}$ such that s is satisfiable in \mathfrak{M} , the conjunction $s \wedge x_{Ds}$ is satisfiable in \mathfrak{M} too. Since we require that the x_{Ds} are isolated, we do not restrict generality by assuming that satisfying models are nice.

Lemma 3.7 is now reformulated as follows.

Lemma 6.1 If \mathfrak{M} is nice for \mathcal{F} , then $S_{\mathfrak{M}}$ is a demo.

Proof

(D \diamond) As in the proof of Lemma 3.7.

(DN) As in the proof of Lemma 3.7.

(DD) Let $Ds \in H_w \in S_{\mathfrak{M}}$. Then there is some $v \neq w$ with $\mathfrak{M}, v \models s$. In case $\mathfrak{M}, w \models x_{Ds}$, we conclude that $x_{Ds} \in H_w$ and $x_{Ds} \notin H_v$. Therefore, $H_v \neq H_w$. Otherwise, since \mathfrak{M} is nice, we can assume w.l.o.g. that v is precisely the state with $\mathfrak{M}, v \models x_{Ds} \wedge s$. This implies $H_v \neq H_w$, too. In both cases, we have $s \in H_v$ from $\mathfrak{M}, v \models s$.

(DD \bar{D}) Let $\bar{D}s \in H_w \in S_{\mathfrak{M}}$, and let $H' \in S_{\mathfrak{M}}$ with $H' \neq H_w$. Hence $H' = H_v$ for some $v \neq w$. From $\mathfrak{M}, w \models \bar{D}s$, we conclude $\mathfrak{M}, v \models s$, i.e., $s \in H_v$. ■

We further need to incorporate the assumptions of models being nice into Lemma 3.8 and its proof (see [19]).

Lemma 6.2 (Demo existence for HPDL $_D$) For every satisfiable formula $s \in \mathcal{F}$, there is a demo \mathcal{D} over \mathcal{F} that contains s .

Proof Let $\mathfrak{M}, w \models s$. Since the auxiliary nominals occur only isolated, we can always make \mathfrak{M} nice for \mathcal{F} by letting the x_{Ds} denote the right states. Therefore we can assume w.l.o.g. that \mathfrak{M} is nice and take $\mathcal{D} = S_{\mathfrak{M}}$, which contains H_w with $s \in H_w$. ■

Extending the proof of Lemma 3.12 is straightforward if we add two cases.

$s = Dt$. If $Dt \in H \in \mathcal{D}$, then Demo Condition (DD) requires the existence of an $H' \in \mathcal{D}$ with $H' \neq H$ and $t \in H'$. The induction hypothesis yields $\mathfrak{M}_{\mathcal{D}}, H' \models t$ and, therefore, $\mathfrak{M}_{\mathcal{D}}, H \models Dt$.

$s = \overline{D}t$. Analogous to the previous case, using Demo Condition (DD \overline{D}).

We conclude that the conceptual cost of adding the existential difference modality is significant, while the technical additions are straightforward once a suitable definition of the auxiliary nominals is in place. The universal difference operator has not caused any difficulties so far, but it will lose its harmlessness when it comes to the decision procedure.

First, we need to incorporate D into the definition of pruning, which is now defined as follows: let $S \xrightarrow{P} S'$ if S' can be obtained from S by deleting some $H \in S$ that violates (D \diamond) or (DD), i.e., one of the following two cases occurs:

1. For some $\langle \alpha \rangle s \in H \in S$, there is no $H' \in S$ such that $H \xrightarrow{\alpha}_S H'$ and $s \in H'$.
2. For some $Ds \in H \in S$, there is no $H' \in S$ such that $H' \neq H$ and $s \in H'$.

Since we have lost the existence of a greatest demo not only because of the auxiliary nominals, but also due to the presence of \overline{D} , we will have to revisit the assumptions of Proposition 5.1 and Theorem 5.2 that say under which conditions pruning respects demos and leads to a unique maximal demo. It suffices to add the requirement that S satisfies (DD \overline{D}) to both assumptions and observe that all subsets of S then satisfy (DD \overline{D}). The proofs then go through unchanged.

Therefore, the decision method from the previous section can be reused if we reformulate Step 3 as follows:

Guess a maximal subset \mathcal{H}' of \mathcal{H} that is nominally coherent and satisfies (DD \overline{D}).

Since \mathcal{H}' needs to be pruned to a demo (candidate) \mathcal{D} , we ensure (DD \overline{D}) by distinguishing three cases for every formula $\overline{D}s \in \mathcal{F}$ about its occurrence in a maximal demo $\mathcal{D} \subseteq \mathcal{H}$.

- (\overline{D} 1) $\overline{D}s$ is not contained in \mathcal{D} . We can therefore discard all Hintikka sets in \mathcal{H} that contain $\overline{D}s$ because they cannot occur in \mathcal{H}' . This ensures that neither \mathcal{H} nor \mathcal{H}' violates (DD \overline{D}) with $\overline{D}s$.
- (\overline{D} 2) All Hintikka sets in \mathcal{D} contain s . Then, for the same reason as above, it is safe to discard all Hintikka sets in \mathcal{H} , ensuring that neither \mathcal{H} nor \mathcal{H}' violates (DD \overline{D}) with $\overline{D}s$.
- (\overline{D} 3) \mathcal{D} contains Hintikka sets H, H' with $\overline{D}s \in H$ and $s \in H'$. If $s \in H$, we are in Case (\overline{D} 2) due to (DD \overline{D}). Hence $s \notin H$, and therefore $\overline{D}s$ is in no Hintikka set other than H . In this case, it is safe to choose one H containing $\overline{D}s$ and not s to remain in \mathcal{H} and \mathcal{H}' , and all other Hintikka sets that contain $\overline{D}s$ or not s can be discarded.

It therefore suffices to add another linear number of guessing actions out of an exponential supply to the guessing phase, and the previous arguments about the determinization in exponential time still apply.

While the addition of D causes significant conceptual overhead, the extension of the decision method with \bar{D} within the given time bounds is non-trivial.

7 Tests

We redefine the transition relation for tests: $\xrightarrow{S} = \{(H, H) \mid s \in H \in S\}$.

In order to prove that the Hintikka system $S_{\mathfrak{M}}$ induced by a model is a demo, it suffices to add one straightforward case to the proof of Lemma 3.6.

$\alpha = t$. If $v \xrightarrow{t}_{\mathfrak{M}} w$, then $v = w$ and $\mathfrak{M}, v \models t$. Then we also have that $H_v = H_w$ and $t \in H_v$. Hence, $H_v \xrightarrow{t}_{S_{\mathfrak{M}}} H_w$.

For the other direction—every demo is satisfied by its induced model—the missing cases in Lemmas 3.10 and 3.11 require additional assumptions, which are added in the following. We restrict the proofs to the cases of tests because the other cases go through unchanged.

Lemma 7.1 Let S be a Hintikka system and α a program such that, for all tests t in α and all $H \in S$, $t \in H$ implies $\mathfrak{M}_S, H \models t$. Then $\xrightarrow{\alpha}_S \subseteq \xrightarrow{\alpha}_{\mathfrak{M}_S}$.

Proof $\alpha = t$. If $H \xrightarrow{t}_S H'$, then $H = H'$ and $t \in H$. From the additional assumption, we obtain $\mathfrak{M}_S, H \models t$ and, therefore, the definition of $\xrightarrow{t}_{\mathfrak{M}_S}$ yields $H \xrightarrow{t}_{\mathfrak{M}_S} H'$. ■

Lemma 7.2 Let S be a Hintikka system with $H, H' \in S$ satisfying

- $[\alpha]s \in H \xrightarrow{\alpha}_{\mathfrak{M}_S} H'$, and
- For all tests t in α and $H \in S$: if $\sim t \in H$, then $\mathfrak{M}_S, H \models \sim t$.

Then $s \in H'$.

Proof $\alpha = t$. Due to the definition of $\xrightarrow{t}_{\mathfrak{M}_S}$, we have $H = H'$ and $\mathfrak{M}_S, H \models t$. Since H is a Hintikka set and $[t]s \in H$, we have that $\sim t \in H$ or $s \in H$. If $\sim t \in H$, then the additional assumption implies $\mathfrak{M}_S, H \models \sim t$, which contradicts $\mathfrak{M}_S, H \models t$. If $s \in H$, the claim is proven because $H = H'$. ■

The formulation of the decision procedure is unaffected by the addition of tests. The only difference in detail is that $\xrightarrow{t}_{\mathcal{H}}$ needs to be computed by cycling through all the exponentially many $H \in \mathcal{H}$ and their linear-size contents.

The surprising consequence is that, even in the presence of tests, a complicated induction order can be avoided in the proofs leading to the demo theorem, provided that the theorem is sufficiently factored out into lemmas.

8 Converse actions

The extension with converse actions is straightforward: in Definition 3.1, we need to replace the case for \xrightarrow{a}_S with

$$\xrightarrow{a}_S = \{(H, H') \mid \text{for all } s : ([a]s \in H \Rightarrow s \in H') \text{ and } ([a^-]s \in H' \Rightarrow s \in H)\},$$

and add the case $\xrightarrow{a^-}_S = \xrightarrow{a^-}_S$, where $\xrightarrow{a^-}_S$ denotes the inverse of \xrightarrow{a}_S . All proofs of Sections 3 and 4 go through after a straightforward converse case has been added in three places.

- Proof of Lemma 3.6

$\alpha = a^-$. We recall that $v \xrightarrow{a^-}_{\mathcal{M}} w$ iff $w \xrightarrow{a}_{\mathcal{M}} v$ and continue with the argument in the case for a .

- Proof of Lemma 3.10

$\alpha = a^-$. We have $\xrightarrow{a^-}_S = \xrightarrow{a^-}_S \subseteq \xrightarrow{a^-}_{\mathcal{M}_S} = \xrightarrow{a^-}_{\mathcal{M}_S}$.

- Proof of Lemma 3.11

$\alpha = a^-$. We can conclude $H' \xrightarrow{a}_{\mathcal{M}_S} H$ and then obtain $s \in H'$ from the definition of $\xrightarrow{a}_{\mathcal{M}_S}$ again.

We conclude that adding converse is conceptually, technically and computationally easy in our setting. The situation is different for tableau-based decision procedures, where converse operators cause significant technical difficulties [2, 14].

9 Related work

We have given a Pratt-style worst-case optimal decision procedure for test-free PDL and step-wise extended it to capture nominals, difference modalities, tests and converse. The correctness of this method is based on transparent proofs of the bounded model theorem (BMT). We now discuss how our approach relates to known approaches of this type.

The basis for our approach is Pratt's work [24], where he sketches a decision method that adds pruning to Fischer and Ladner's method [10], without a formal correctness proof. This is probably the most straightforward way of obtaining the exponential-time upper bound. We have extended Pratt's method with hybrid operators and converse.

Variations of Pratt's straightforward approach are given in the literature [16, 22, 17, 3]. In order to establish the BMT, the authors of [16, 22, 17] use either

simultaneous induction over formulas and programs, or a non-standard induction over an order on formulas with and without flags. The correctness of the decision procedure is not immediate, and requires to partially repeat the course of the proof of the BMT. The procedure in [3, Section 6.8] requires Hintikka sets to be upwards saturated, and its proofs use a simpler induction scheme. However, that approach does not include tests. In contrast to [16, 22, 17], our technique factorizes the BMT in a way that every lemma can be proven using a single induction over the subterm relation. The demo notion and its properties can be directly used to conclude the correctness of the decision procedure.

In [23], the authors state that HPDL^- has the same deterministic upper bound as PDL without giving an explicit decision procedure. The work in [5, 1] establishes a chain of polynomial-time satisfiability-preserving translations $\text{HPDL}_{\bar{D}}^- \rightarrow \text{HPDL}_{\bar{E}}^- \rightarrow \text{HPDL}^- \rightarrow \text{PDL}^- \rightarrow \text{PDL}$, again without an explicit decision procedure.

In contrast to the work in [25, 13, 14, 20], our approach does not build a tableau in `Stage construct`, and therefore the edges of the transition relation do not need to be an explicit part of our Hintikka systems. The approach in [20] uses the notion of a clause instead of a Hintikka set and a support relation between clauses and formulas. The support closures of clauses—i.e., their supported sets of formulas—are exactly the Hintikka sets. Without the need for computational optimizations, Hintikka sets are sufficient for our approach and more convenient. While the decision procedure in [20] runs in NEXPTIME for nominals, we show how to obtain an EXPTIME procedure, answering the question from [20] whether a demo can be found efficiently w.r.t. the size of the tableau.

In [14], a more practical worst-case optimal decision method for PDL^- is given and implemented. In contrast, we have acknowledged above that our procedure is not implementable, and this is not the purpose of this paper. We have examined how Pratt’s most simple optimal procedure scales to more expressive features and, in our point of view, this serves the understanding of this type of procedure rather than its implementation.

Syntactic descriptions of models related to demos can be found in [8]. Their Hintikka structures for CTL are richer: they contain the transition relation explicitly and may contain several copies of a Hintikka set.

For future work, it would be interesting to examine whether our approach can be extended to the hybrid μ -calculus and whether graded modalities can be incorporated. On the more practical side, a natural next step is to study how to transfer our approach into a decision procedure that interleaves building a (graph) tableau and pruning in the presence of hybrid operators.

Acknowledgments. We thank the anonymous reviewers for helpful comments.

References

- [1] Carlos Areces, Patrick Blackburn, and Maarten Marx. The computational complexity of hybrid temporal logics. *L. J. IGPL*, 8(5):653–679, 2000.
- [2] Franz Baader and Uli Sattler. Tableau algorithms for description logics. In R. Dyckhoff, editor, *Proc. TABLEAUX 2000*, volume 1847 of *LNCS*, pages 1–18. Springer, 2000.
- [3] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Cambridge University Press, 2001.
- [4] Thomas Bolander and Patrick Blackburn. Termination for hybrid tableaux. *J. Log. Comput.*, 17(3):517–554, 2007.
- [5] Giuseppe De Giacomo. *Decidability of class-based knowledge representation formalisms*. PhD thesis, Università degli Studi di Roma “La Sapienza”, 1995.
- [6] Giuseppe De Giacomo and Fabio Massacci. Combining deduction and model checking into tableaux and algorithms for converse-PDL. *Inf. Comput.*, 162(1-2):117–137, 2000.
- [7] Francesco M. Donini and Fabio Massacci. EXPTIME tableaux for \mathcal{ALC} . *Artif. Intell.*, 124(1):87–138, 2000.
- [8] E. Allen Emerson and Joseph Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *J. Comput. System Sci.*, 30(1):1–24, 1985.
- [9] Michael J. Fischer and Richard E. Ladner. Propositional modal logic of programs. In *Proc. STOC*, pages 286–294. ACM, 1977.
- [10] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *J. Comput. System Sci.*, pages 194–211, 1979.
- [11] George Gargov and Valentin Goranko. Modal logic with names. *J. Philos. Log.*, 22:607–636, 1993.
- [12] Rajeev Goré and Linh Anh Nguyen. EXPTIME tableaux with global caching for description logics with transitive roles, inverse roles and role hierarchies. In *TABLEAUX’07*, volume 4548 of *LNCS*, pages 133–148. Springer, 2007.
- [13] Rajeev Goré and Florian Widmann. An optimal on-the-fly tableau-based decision procedure for PDL-satisfiability. In *CADE-22*, volume 5663 of *LNCS*, pages 437–452. Springer, 2009.

- [14] Rajeev Goré and Florian Widmann. Optimal and cut-free tableaux for propositional dynamic logic with converse. In *IJCAR 2010*, volume 6173 of *LNCS*, pages 225–239. Springer, 2010.
- [15] Volker Haarslev and Ralf Möller. RACER system description. In *IJCAR 2001*, volume 2083 of *LNCS*, pages 701–705. Springer, 2001.
- [16] David Harel. Dynamic logic. In Dov Gabbay and Franz Guenther, editors, *Handbook of Philosophical Logic*, volume II, pages 497–604. Reidel, 1984.
- [17] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic Logic*. The MIT Press, 2000.
- [18] Bernhard Hollunder, Werner Nutt, and Manfred Schmidt-Schauß. Subsumption algorithms for concept description languages. In *Proc. ECAI*, pages 348–353, 1990.
- [19] Mark Kaminski, Thomas Schneider, and Gert Smolka. Correctness and worst-case optimality of Pratt-style decision procedures for modal and hybrid logics. Technical report, Saarland University, 2011.
- [20] Mark Kaminski and Gert Smolka. Clausal graph tableaux for hybrid logic with eventualities and difference. In *Proc. LPAR-17*, volume 6397 of *LNCS*, pages 417–431. Springer, 2010.
- [21] Mark Kaminski and Gert Smolka. Terminating tableaux for hybrid logic with eventualities. In *Proc. IJCAR*, volume 6173 of *LNCS*, pages 240–254. Springer, 2010.
- [22] Dexter Kozen and Jerzy Tiuryn. Logics of programs. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 789–840. Elsevier, 1990.
- [23] Solomon Passy and Tinko Tinchev. An essay in combinatory dynamic logic. *Inf. Comput.*, 93(2):263–332, 1991.
- [24] Vaughan R. Pratt. Models of program logics. In *Proc. FOCS*, pages 115–122. IEEE Computer Society Press, 1979.
- [25] Vaughan R. Pratt. A near-optimal method for reasoning about action. *J. Comput. System Sci.*, 20(2):231–254, 1980.
- [26] Arthur Prior. *Past, Present and Future*. OUP, 1967.
- [27] Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with compliments. *Artif. Intell.*, 48(1):1–26, 1991.

- [28] Krister Segerberg. A note on the logic of elsewhere. *Theoria*, 46(2-3):183-187, 1980.
- [29] Robert S. Streett and E. Allen Emerson. An automata theoretic decision procedure for the propositional μ -calculus. *Inform. and Control*, 81:249-264, 1989.
- [30] Dmitry Tsarkov, Ian Horrocks, and Peter F. Patel-Schneider. Optimizing terminological reasoning for expressive description logics. *J. Autom. Reasoning*, 39(3):277-316, 2007.
- [31] Moshe Y. Vardi and Pierre Wolper. Automata-theoretic techniques for modal logics of programs. *J. Comput. System Sci.*, 32:183-221, 1986.