PDL with Intersection and Converse is 2EXP-complete

Stefan Göller[1], Markus Lohrey[1]*, and Carsten Lutz[2]

[1] Universität Stuttgart, FMI, Germany
[2] Institute for Theoretical Computer Science, TU Dresden, Germany
goeller,lohrey@informatik.uni-stuttgart.de,
lutz@tcs.inf.tu-dresden.de

Abstract. We study the complexity of satisfiability for the expressive extension ICPDL of PDL (Propositional Dynamic Logic), which admits intersection and converse as program operations. Our main result is containment in 2EXP, which improves the previously known non-elementary upper bound and implies 2EXP-completeness due to an existing lower bound for PDL with intersection. The proof proceeds by showing that every satisfiable ICPDL formula has a model of tree-width at most two and then giving a reduction to the emptiness problem for alternating two-way automata on infinite trees. In this way, we also reprove in an elegant way Danecki's difficult result that satisfiability for PDL with intersection is in 2EXP.

1 Introduction

Propositional Dynamic Logic (PDL) was introduced by Fischer and Ladner in 1979 as a modal logic for reasoning about the input/output behaviour of programs [6]. In PDL, there are two syntactic entities: formulas, built from Boolean and modal operators and interpreted as sets of nodes of a Kripke structure; and programs, built from the operators test, union, composition, and Kleene star (reflexive transitive closure) and interpreted as binary relations in a Kripke structure. Since its invention, many different extensions of PDL have been proposed, mainly by allowing additional operators on programs. Three of the most prominent extensions are PDL with the converse operator (CPDL), PDL with the intersection operator (IPDL), and PDL with the negation operator on programs (NPDL), see the monograph [9] and references therein. While some of these extensions such as CPDL are well-suited for reasoning about programs, most of them aim at the numerous other applications that PDL has found since its invention. Notable examples of such applications include agent-based systems [13], regular path constraints [2], and XML-querying [1, 16]. In AI, PDL received attention due to its close relationship to description logics [7] and epistemic logic [17].

The most important decision problem for PDL is satisfiability: is there a Kripke structure which satisfies a given formula at some node? A classical result of Fischer and Ladner states that satisfiability for PDL is EXP-complete [6, 15]. The EXP upper bound extends without difficulty to CPDL and can even be established for several extensions of CPDL [18]. In contrast, the precise complexity of satisfiability for IPDL was a long

* The first and second are supported by the DFG project GELO.

standing open problem. In [4], Danecki proved a 2EXP upper bound. Alas, Danecki's proof is rather difficult and many details are omitted in the published version. One of the reasons for the difficulty of IPDL is that, unlike PDL, it lacks the tree model property, i.e., a satisfiable IPDL formula does not necessarily have a tree model. Danecki proved that every satisfiable IPDL formula has a special model which can be encoded by a tree. This observation paves the way to using automata theoretic techniques in decision procedures for IPDL. Only recently, a matching 2EXP lower bound for IPDL was shown by Lange and the third author [10]. Regarding NPDL, it is long known that satisfiability is undecidable [9]. As recently shown in [9], the fragment of NPDL, where program negation is restricted to atomic programs, is decidable and EXP-complete.

In this paper, we consider extensions of PDL with (at least two of) converse, intersection, and negation. Our main result concerns the complexity of satisfiability in ICPDL, the extension of PDL with both converse and intersection. Decidability was shown by the third author in [11] using a reduction to monadic second order logic over the infinite binary tree. However, this only yields a nonelementary algorithm which does not match the 2EXP lower bound that ICPDL inherits from IPDL. We prove that satisfiability in ICPDL can be decided in 2EXP, and thus settle the complexity of ICPDL as 2EXP-complete. There are some additional virtues of our result. First, we provide a shorter and (hopefully) more comprehensible proof of the 2EXP upper bound for IPDL. Second, the information logic DAL (data analysis logic) [5] is a fragment of ICPDL (but not of IPDL) and thus inherits the 2EXP upper bound. And third, our result has applications in description logic and epistemic logic, see [11] for more details.

Our main result is proved in three clearly separated parts. In part one, we establish a certain model property for ICPDL based on the notion of tree width. Tree width measures how close a graph is to a tree, and is one of the most important concepts in modern graph theory with many applications in computer science. As mentioned earlier, IPDL (and hence also ICPDL) does not have the tree model property. We prove that ICPDL enjoys an "almost tree model property": every satisfiable ICPDL formula has a model of tree width at most two. This part of our proof is comparable to Danecki's observation that every satisfiable IPDL formula has a special model which can be encoded by a tree.

In part two of our proof, we use the established model property to give a polytime reduction of satisfiability in ICPDL to what we call *ω-regular tree satisfiability* in ICPDL. The latter problem is defined in terms of two-way alternating parity tree automata (TWAPTAs). A TWAPTA is an alternating automaton that runs on infinite node-labeled trees and has the possibility to move upwards and downwards in the tree. Acceptance is defined via a parity condition. Infinite node-labeled trees can be viewed in a natural way as Kripke structures and thus we can interprete ICPDL formulas in such trees. Now, ω-regular tree satisfiability in ICPDL is the following problem: given an ICPDL formula φ and a TWAPTA $\mathcal{T}$, is there a tree accepted by $\mathcal{T}$ which is a model for φ? Our reduction of satisfiability in ICPDL to this problem is based on a suitable encoding of width two tree decompositions of Kripke structures. The TWAPTA constructed in the reduction accepts precisely such encodings.

Finally, in part three we reduce ω-regular tree satisfiability in ICPDL to the emptiness problem for TWAPTAs. The latter problem was shown to be EXP-complete in [19]. Since our reduction of ω-regular tree satisfiability in ICPDL to TWAPTA-emptiness involves an exponential blow-up in automata size, we obtain an 2EXP upper bound for ω-regular tree satisfiability in ICPDL and also for standard satisfiability in ICPDL. The reduction employs a technique from [8], where the first and second author proved that the model-checking problem for IPDL over transition graphs of pushdown automata is 2EXP-complete. In fact, this model-checking problem can be easily reduced to ω-regular tree satisfiability in ICPDL. This illustrates that ω-regular tree satisfiability in ICPDL is of interest beyond its application in the current paper.

To obtain a more complete picture, we also investigate the option of extending ICPDL with program negation. It turns out that in the presence of intersection, program negation is problematic from a computational perspective. In particular, we prove that already IPDL extended with negation restricted to atomic programs is undecidable. This should be contrasted with the decidability result for PDL extended with atomic negation mentioned above [12]. Missing proofs can be found in the appendix.

2 ICPDL

Let $\mathbb{P}$ be a set of *atomic propositions* and $\mathbb{A}$ a set of *atomic programs*. *Formulas* φ and *programs* π of the logic ICPDL are defined by the following grammar, where p ranges over $\mathbb{P}$ and a over $\mathbb{A}$:

$$\begin{aligned}\varphi &::= p \mid \neg\varphi \mid \langle\pi\rangle\,\varphi\\ \pi &::= a \mid \pi_1 \cup \pi_2 \mid \pi_1 \cap \pi_2 \mid \pi_1 \circ \pi_2 \mid \pi^* \mid \overline{\pi} \mid \varphi?\end{aligned}$$

We introduce the usual abbreviations $\varphi_1 \wedge \varphi_2 = \langle\varphi_1?\rangle\varphi_2$, $\varphi_1 \vee \varphi_2 = \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, and $[\pi]\varphi = \neg\langle\pi\rangle\neg\varphi$. The fragment IPDL of ICPDL is obtained by dropping the $\overline{\pi}$ clause from the above grammar.

The *semantics* of ICPDL is defined in terms of Kripke structures. A *Kripke structure* is a tuple $K = (X, \{\rightarrow_a \mid a \in \mathbb{A}\}, \rho)$, where (i) X is a set of *states*, (ii) $\rightarrow_a \subseteq X \times X$ is a *transition relation* for each $a \in \mathbb{A}$, and (iii) $\rho : X \to 2^{\mathbb{P}}$ assigns to each state a set of atomic propositions. Given a Kripke structure $K = (X, \{\rightarrow_a \mid a \in \mathbb{A}\}, \rho)$, we define by mutual induction for each ICPDL program π a binary relation $[\![\pi]\!]_K \subseteq X \times X$ and for each ICPDL formula φ a subset $[\![\varphi]\!]_K \subseteq X$ as follows ($\circ$ denotes the composition operator for binary relations: $R \circ S = \{(a,b) \mid \exists c : (a,c) \in R, (c,b) \in S\}$):

$$\begin{aligned}
[\![p]\!]_K &= \{x \mid p \in \rho(x)\} \text{ for } p \in \mathbb{P}\\
[\![\neg\varphi]\!]_K &= X \setminus [\![\varphi]\!]_K\\
[\![\langle\pi\rangle\varphi]\!]_K &= \{x \mid \exists y : (x,y) \in [\![\pi]\!]_K \wedge y \in [\![\varphi]\!]_K\}\\
[\![a]\!]_K &= \rightarrow_a \text{ for } a \in \mathbb{A}\\
[\![\varphi?]\!]_K &= \{(x,x) \mid x \in [\![\varphi]\!]_K\}\\
[\![\pi^*]\!]_K &= [\![\pi]\!]_K^*\\
[\![\overline{\pi}]\!]_K &= \{(y,x) \mid (x,y) \in [\![\pi]\!]_K\}\\
[\![\pi_1 \text{ op } \pi_2]\!]_K &= [\![\pi_1]\!]_K \text{ op } [\![\pi_2]\!]_K \text{ for op} \in \{\cup, \cap, \circ\}
\end{aligned}$$

For $x \in X$ we write $(K,x) \models \varphi$ if $x \in [\![\varphi]\!]_K$. If $(K,x) \models \varphi$ for some $x \in X$, then K is a *model* of φ. The formula φ *satisfiable* if there exists some model for φ.

Since the converse operator can be pushed down to atomic programs, we assume for the rest of this paper that converse is only applied to atomic programs. Let us set $\overline{\mathbb{A}} = \{\overline{a} \mid a \in \mathbb{A}\}$. The size $|\varphi|$ of an ICPDL formula φ and the size $|\pi|$ of an ICPDL program π is defined as follows: $|p| = |a| = 1$ for all $p \in \mathbb{P}$ and $a \in \mathbb{A} \cup \overline{\mathbb{A}}$, $|\neg\varphi| = |\varphi?| = |\varphi| + 1$, $|\langle\pi\rangle\varphi| = |\pi| + |\varphi|$, $|\pi_1 \text{ op } \pi_2| = |\pi_1| + |\pi_2| + 1$ for op $\in \{\cup, \cap, \circ\}$, and $|\pi^*| = |\pi| + 1$.

The main result of this paper is the following.

Theorem 1. *Satisfiability in ICPDL is* $2\mathsf{EXP}$*-complete.*

As discussed in the introduction, it suffices to give a $2\mathsf{EXP}$ algorithm for satisfiability in ICPDL because of the known $2\mathsf{EXP}$ lower bound for IPDL [10]. The rest of the paper is organized as follows. In Section 3, we show that every satisfiable ICPDL formula has a model of tree width at most two. In Section 4, satisfiability of ICPDL formulas in a model of tree width at most two is reduced to ω-regular tree satisfiability in ICPDL. In Section 6, the latter problem is shown to be in $2\mathsf{EXP}$. Finally, Section 7 contains the undecidability proof for IPDL extended with negation of atomic programs.

3 Models of Tree-Width Two Suffice

We start with defining the tree-width of Kripke structures. For technical reasons, we consider only countable structures in this context. As will become clear later, this can be done w.l.o.g. Let $K = (X, \{\rightarrow_a \mid a \in \mathbb{A}\}, \rho)$ be a countable Kripke structure. A *tree decomposition* of K is a tuple $(T, (X_v)_{v \in V})$, where $T = (V, E)$ is a countable undirected tree, X_v is a subset of X (also called a *bag*) for all $v \in V$, and the following conditions are satisfied:

- $\bigcup_{v \in V} X_v = X$
- For every transition $x \rightarrow_a y$ of K there exists $v \in V$ with $x, y \in X_v$.
- For every $x \in X$, the set $\{v \in V \mid x \in X_v\}$ is a connected subset of the tree T.

The width of this tree decomposition is the supremum of $\{|X_v| - 1 \mid v \in V\}$. The *tree width* of a Kripke structure K is the minimal k such that K has a tree decomposition of width k. The purpose of this section is to prove the following theorem.

Theorem 2. *Every satisfiable ICPDL formula has a countable model of tree width at most two.*

As a preliminary to proving Theorem 2, we mutually define the set of *subprograms* $\mathsf{subp}(\alpha)$ and the set of *subformulas* $\mathsf{subf}(\alpha)$, where α is either an ICPDL formula or an ICPDL program:

- $\mathsf{subp}(a) = \{a\}$, $\mathsf{subp}(\overline{a}) = \{a, \overline{a}\}$, $\mathsf{subf}(a) = \mathsf{supf}(\overline{a}) = \emptyset$ for $a \in \mathbb{A}$;
- $\mathsf{subp}(\pi) = \{\pi\} \cup \mathsf{subp}(\pi_1) \cup \mathsf{subp}(\pi_2)$ and $\mathsf{subf}(\pi) = \mathsf{subf}(\pi_1) \cup \mathsf{subf}(\pi_2)$ if $\pi = \pi_1 \text{ op } \pi_2$ for op $\in \{\cup, \cap, \circ\}$;
- $\mathsf{subp}(\pi^*) = \{\pi^*\} \cup \mathsf{subp}(\pi)$ and $\mathsf{subf}(\pi^*) = \mathsf{subf}(\pi)$;
- $\mathsf{subp}(\varphi?) = \{\varphi?\} \cup \mathsf{subp}(\varphi)$ and $\mathsf{subf}(\varphi?) = \mathsf{subf}(\varphi)$
- $\mathsf{subp}(p) = \emptyset$ and $\mathsf{subf}(p) = \{p\}$ for $p \in \mathbb{P}$;

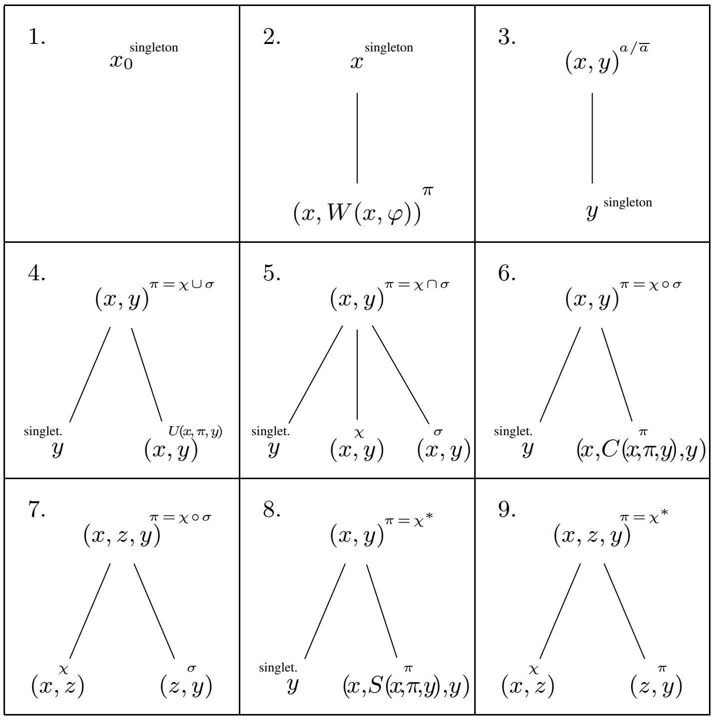

Fig. 1. Inductive definition of $(T, (t_v)_{v\in V})$.

- $\mathsf{subp}(\neg\varphi) = \mathsf{subp}(\varphi)$ and $\mathsf{subf}(\neg\varphi) = \{\neg\varphi\} \cup \mathsf{subf}(\varphi)$;
- $\mathsf{subp}(\langle\pi\rangle\varphi) = \mathsf{subp}(\pi) \cup \mathsf{subp}(\varphi)$ and $\mathsf{subf}(\langle\pi\rangle\varphi) = \{\langle\pi\rangle\varphi\} \cup \mathsf{subf}(\pi) \cup \mathsf{subf}(\varphi)$.

To prove Theorem 2, fix a satisfiable ICPDL formula φ_0, a model $K = (X, \{\rightarrow_a \mid a \in \mathbb{A}\}, \rho)$ of φ_0, and a state $x_0 \in [\![\varphi_0]\!]_K$. Moreover, fix choice functions W, U, C, and S such that

- if $\varphi = \langle\pi\rangle\psi \in \mathsf{subf}(\varphi_0)$ and $x \in [\![\varphi]\!]_K$, then $W(x,\varphi) = y \in X$ such that $y \in [\![\psi]\!]_K$ and $(x,y) \in [\![\pi]\!]_K$;
- if $\pi = \chi \cup \sigma \in \mathsf{subp}(\varphi_0)$ and $(x,y) \in [\![\pi]\!]_K$, then $U(x,\pi,y) = \tau \in \{\chi,\sigma\}$ such that $(x,y) \in [\![\tau]\!]_K$.
- if $\pi = \chi \circ \sigma \in \mathsf{subp}(\varphi_0)$ and $(x,y) \in [\![\pi]\!]_K$, then $C(x,\pi,y) = z \in X$ such that $(x,z) \in [\![\chi]\!]_K$ and $(z,y) \in [\![\sigma]\!]_K$;
- if $\pi = \chi^* \in \mathsf{subp}(\varphi_0)$ and $(x,y) \in [\![\pi]\!]_K$ with $x \neq y$, then $S(x,\pi,y) = z \in X$ such that there exists a sequence $x_0,\ldots,x_n \in X$ with
  1. $x_0 = x$ and $x_n = y$;
  2. $(x_i, x_{i+1}) \in [\![\chi]\!]_K$ for all $i < n$;
  3. $x_0,\ldots,x_n$ is a shortest sequence with Properties 1 and 2;
  4. $x_1 = z$.

Now we inductively define a node-labeled tree $(T, (t_v)_{v \in V})$ with $T = (V, E)$ and $t_v \in X \cup X^2 \cup X^3$ for all $v \in V$. During the construction, each node in the tree is assigned a type, which may either be "singleton" or π for $\pi \in \mathsf{subp}(\varphi_0)$. Figure 1 illustrates the different cases, which are as follows:

1. Start the construction with a root node v of type singleton and set $t_v = x_0$;
2. if $v \in V$ is of type singleton and $t_v = x$, then for every $\varphi = \langle \pi \rangle \psi \in \mathsf{subf}(\varphi_0)$ such that $x \in [\![\varphi]\!]_K$, add a successor w of type π and set $t_w = (x, W(x, \varphi))$;
3. if $v \in V$ is of type a or $\overline{a}$, where $a \in \mathbb{A}$ and $t_v = (x, y)$, then add a successor w of type singleton and set $t_w = y$;
4. if $v \in V$ is of type $\pi = \chi \cup \sigma$ and $t_v = (x, y)$, then
   - add a successor w of type singleton and set $t_w = y$;
   - add a successor w' of type $U(x, \pi, y)$ and set $t_{w'} = (x, y)$;
5. if $v \in V$ is of type $\pi = \chi \cap \sigma$ and $t_v = (x, y)$, then
   - add a successor w of type singleton and set $t_w = y$;
   - add successors u, u' of type χ and σ, respectively, and set $t_u = t_{u'} = (x, y)$;
6. if $v \in V$ is of type $\pi = \chi \circ \sigma$ and $t_v = (x, y)$, then
   - add a successor w of type singleton and set $t_w = y$;
   - add a successor w' of type π and set $t_w = (x, C(x, \pi, y), y)$;
7. if $v \in V$ is of type $\pi = \chi \circ \sigma$ and $t_v = (x, z, y)$, then add successors u, u' of type χ and σ and set $t_u = (x, z)$ and $t_{u'} = (z, y)$;
8. if $v \in V$ is of type $\pi = \chi^*$ and $t_v = (x, y)$ with $x \neq y$, then
   - add a successor w of type singleton and set $t_w = y$;
   - add a successor w' of type π and set $t_w = (x, S(x, \pi, y), y)$;
9. if $v \in V$ is of type $\pi = \chi^*$ and $t_v = (x, z, y)$, then add successors u, u' of type χ and π, respectively, and set $t_u = (x, z)$ and $t_{u'} = (z, y)$.

We assume that successors are added at most once to each node in the induction step and that the construction proceeds in a breadth first manner. Note that nodes of type $\psi?$ are always leafs, and so are nodes v of type χ^* with $t_v = (x, x)$ for some $x \in X$. Another important property, which illustrates the connection between K and the constructed tree, is the following:

$$\forall v \in V : \text{if } v \text{ is of type } \pi \text{ and } t_v = (x, y), \text{ then } (x, y) \in [\![\pi]\!]_K. \qquad (\dagger)$$

A *place* is a pair (v, x) such that x is a member of t_v. We denote the set of all places with P and let $\sim$ be the smallest equivalence relation on P which contains all pairs of the form $((u, x), (v, x))$, where $(u, v) \in E$ is an edge of the tree T. We use $[v, x]$ to denote the equivalence class of $(v, x) \in P$ w.r.t. the relation $\sim$. Define a Kripke structure $K' = (X', \{\to'_a \mid a \in \mathbb{A}\}, \rho')$ as follows:

- $X' = \{[v, x] \mid (v, x) \in P\}$;
- $[v, x] \to'_a [v', y]$ if and only if at least one of the following holds:
  - there is $u \in V$ of type a s.t. $t_u = (x, y)$, $(u, x) \sim (v, x)$, and $(u, y) \sim (v', y)$;
  - there is $u \in V$ of type $\overline{a}$ s.t. $t_u = (y, x)$, $(u, x) \sim (v, x)$, and $(u, y) \sim (v', y)$.
- $\rho'([v, x]) = \rho(x)$.

Since K' is clearly countable, to finish the proof it suffices to show the following:

1. setting $X_v = \{[v,x] \mid x \text{ occurs in } t_v\}$ for all $v \in V$, we obtain a tree decomposition $(T, (X_v)_{v \in V})$ of K' of width two;
2. K' satisfies φ_0.

Using the definitions of K' and $\sim$, it is readily checked that $(T, (X_v)_{v \in V})$ is a tree decomposition of K'. Tree width two is then immediate by construction of $(T, (t_v)_{v \in V})$. Finally, we can prove the following, whose Point 3 yields that K' is a model of φ_0.

Lemma 1. *For all $v, u \in V$, $x, y \in X$, $\pi \in \mathsf{subp}(\varphi_0)$, and $\varphi \in \mathsf{subf}(\varphi_0)$,*

1. *if $t_v = (x,y)$ and v is of type π, then $([v,x],[v,y]) \in [\![\pi]\!]_{K'}$;*
2. *if $(v,x),(u,y) \in P$ and $([v,x],[u,y]) \in [\![\pi]\!]_{K'}$, then $(x,y) \in [\![\pi]\!]_K$;*
3. *if $(v,x) \in P$, then $(K,x) \models \varphi$ if and only if $(K',[v,x]) \models \varphi$.*

4 Reduction to ω-Regular Tree Satisfiability

We exploit the model property established in the previous section to reduce satisfiability in ICPDL to ω-regular tree satisfiability in ICPDL. Since the latter is defined in terms of alternating automata on infinite trees, we start with introducing these automata and the trees on which they work.

Let Γ and Υ be finite sets. A Γ-labeled (directed) Υ-tree is a partial function $T : \Upsilon^* \to \Gamma$ such that $\mathrm{dom}(T)$ (the set of nodes) is prefix-closed. If $\mathrm{dom}(T) = \Upsilon^*$, then T is called *complete*. If Υ is understood or not important, we simply talk of Γ-labeled trees. We deliberately work with two kinds of trees here: undirected trees as a basis for tree decompositions in Section 3, and directed trees introduced here as the objects on which alternating tree automata work.

Let P be a finite set of atomic propositions and A a finite set of atomic programs, not necessarily identical to the sets $\mathbb{P}$ and $\mathbb{A}$ fixed in Section 2. A complete 2^{P}-labeled A-tree T can be viewed as a Kripke structure $K_T = (\mathsf{A}^*, \{\to_a \mid a \in \mathsf{A}\}, T)$ over the set of atomic propositions P and atomic programs A, where $\to_a = \{(u, ua) \mid u \in \mathsf{A}^*\}$ for all $a \in \mathsf{A}$. In the following, we identify T and the associated Kripke structure K_T.

We now define alternating automata on complete Γ-labeled Υ-trees. For a finite set X we denote by $\mathcal{B}^+(X)$ the set of all *positive boolean formulas* with elements of X used as variables. The constants `true` and `false` are admitted. A subset $Y \subseteq X$ can be seen as a valuation in the obvious way: it *satisfies* a formula $\theta \in \mathcal{B}^+(X)$ if and only if by assigning `true` to all elements in Y the formula θ is evaluated to `true`. Define the set of Υ-*moves* as $\mathrm{mov}(\Upsilon) = \Upsilon \uplus \overline{\Upsilon} \uplus \{\varepsilon\}$, where $\overline{\Upsilon} = \{\overline{a} \mid a \in \Upsilon\}$. For $u \in \Upsilon^*$ and $a \in \Upsilon$, define $u\overline{a} = v$ if $u = va$ for some $v \in \Upsilon^*$ and $u\overline{a} =$ undefined if $u \notin \Upsilon^* a$. A *two-way alternating parity tree automaton* (TWAPTA for short) over Γ-labeled Υ-trees is a tuple $\mathcal{T} = (S, \delta, s_0, \mathrm{Acc})$, where (i) S is a finite non-empty set of states, (ii) $\delta : S \times \Gamma \to \mathcal{B}^+(S \times \mathrm{mov}(\Upsilon))$ is the *transition function*, (iii) $s_0 \in S$ is the *initial state*, and (iv) $\mathrm{Acc} : S \to \{0, \ldots, m\}$ is the *priority function* (where $m \in \mathbb{N}$) which assigns to each state an integer between 0 and m. Define $|\mathrm{Acc}| = \max\{\mathrm{Acc}(s) \mid s \in S\}$. Let T a complete Γ-labeled Υ-tree, $u \in \Upsilon^*$ a node, and $s \in S$ a state. An $(s,u)_T$-*run* of $\mathcal{T}$ is a (not necessarily complete) $(S \times \Upsilon^*)$-labeled Ω-tree T_R for some set finite Ω such that the following two conditions are satisfied: (i) $T_R(\varepsilon) = (s,u)$, and (ii) if

$\alpha \in \mathrm{dom}(T_R)$ with $T_R(\alpha) = (q, v)$ and $\delta(q, T(v)) = \theta$, then there exists a subset $Y \subseteq S \times \mathrm{mov}(\Upsilon)$ that satisfies the formula θ and for all $(s', e) \in Y$: ve is defined and there exists an $\omega \in \Omega$ with $\alpha\omega \in \mathrm{dom}(T_R)$ and $T_R(\alpha\omega) = (s', ve)$. We say that an $(s, u)_T$-run is *successful*, if for every infinite path $\alpha_1\alpha_2 \cdots \in \mathrm{dom}(T_R)^\omega$ of T_R ($\alpha_1 = \varepsilon$, $\alpha_{i+1} = \alpha_i\omega$ for some $\omega \in \Omega$) the number $\min\{\mathrm{Acc}(q) \mid q \in S, T_R(\alpha_i) \in \{q\} \times \Upsilon^*$ for infinitely many $i\}$ is even. Define

$$\begin{aligned} [\![\mathcal{T}, s]\!]_T &= \{u \in \Upsilon^* \mid \text{ there exists a successful } (s, u)_T\text{-run of } \mathcal{T}\} \\ L(\mathcal{T}) &= \{T \mid \varepsilon \in [\![\mathcal{T}, s_0]\!]_T\} \end{aligned}$$

The subscript T is omitted if clear from the context. An *ω-regular tree language* L is a set of complete Γ-labeled Υ-trees such that $L(\mathcal{T}) = L$ for some TWAPTA $\mathcal{T}$.

Our TWAPTA model differs slightly from other definitions in the literature: First, we run TWAPTA only on complete trees; this will be convenient in Section 5 and 6. Second, usually a TWAPTA has an operation $\uparrow$ for moving to the parent node of the current node. In our model, $\uparrow$ is replaced by the operations $\overline{a} \in \overline{\Upsilon}$ for all $a \in \Upsilon$. The operation $\overline{a}$ can only be executed if the current node is an a-successor of its parent node. It is easy to see that these two models are equivalent.

In Section 6, we will make use of the following result of Vardi:

Theorem 3 ([19]). *For a given TWAPTA $\mathcal{T} = (Q, \delta, s_0 \mathrm{Acc})$ it can be checked in time exponential in $|Q| \cdot |\mathrm{Acc}|$ whether $L(\mathcal{T}) = \emptyset$.*

We are now in the position to formally define *ω-regular tree satisfiability in ICPDL*: given a TWAPTA $\mathcal{T}$ over 2^{P}-labeled A-trees and an ICPDL formula φ over the set of atomic propositions P and set of atomic programs A (in the following we simply say *over* P *and* A), decide whether there is a $T \in L(\mathcal{T})$ such that $(T, \varepsilon) \models \varphi$.

To reduce satisfiability in ICPDL to ω-regular tree satisfiability in ICPDL, we translate an ICPDL formula φ over $\mathbb{P}$ and $\mathbb{A}$ into a TWAPTA $\mathcal{T}$ and an ICPDL formula $\widehat{\varphi}$ over

$$\mathsf{A} = \{a, b, 0, 1, 2\} \quad \text{and} \quad \mathsf{P} = \{t\} \cup \mathsf{prop}(\varphi) \cup (\{0, 1, 2\} \times \mathsf{prog}(\varphi) \times \{0, 1, 2\}),$$

where $\mathsf{prop}(\varphi) = \mathsf{subf}(\varphi) \cap \mathbb{P}$ and $\mathsf{prog}(\varphi) = \mathsf{suba}(\varphi) \cap \mathbb{A}$. Intuitively, each 2^{P}-labeled A-tree T accepted by $\mathcal{T}$ encodes a tree decomposition of a Kripke structure K over $\mathbb{P}$ and $\mathbb{A}$ of tree width at most two (in a sense yet to be made precise), and T is a model of $\widehat{\varphi}$ if and only if K is a model of φ. To achieve an elegant encoding of tree decompositions, we work with *good* tree decompositions. A tree decomposition $(T, (X_v)_{v \in V})$ with $T = (V, E)$ is called good if

- $V = \{a, b\}^*$, i.e., T is a complete binary tree, and
- $X_v \subseteq X_{vc}$ or $X_{vc} \subseteq X_v$ for all $v \in V$ and $c \in \{a, b\}$.

It is easily seen how to convert a tree decomposition of a Kripke structure K of width k into a good tree decomposition of K of width k by introducing additional nodes.

Lemma 2. *Every countable Kripke structure of tree width k has a good tree decomposition of width k.*

In the following we assume that $k = 2$, since this is the only interesting case in this paper. To encode a Kripke structure together with a good tree decomposition $(T, (X_v)_{v \in V})$ of width at most two as a 2^{P}-labeled A-tree, we think of every tree node $v \in \{a, b\}^*$ as being divided into three slots which can be empty or filled with a state of the Kripke structure. When moving to a child, by the second condition of good tree decompositions we either add nodes to empty slots or remove nodes from slots, but not both. The three slots of the node v are described by new leafs $v0, v1, v2$. This explains our choice of A above. When slot vi is occupied by a state of the Kripke structure, then vi receives the special label $t \in \mathsf{P}$. Finally, information about the edges of the Kripke structure are stored in tree nodes from $\{a, b\}^*$. We now formally define these encodings: a complete 2^{P}-labeled A-tree T is called *valid* if the following holds for all $v \in \mathsf{A}^*$:

- if $v \in \{a, b\}^*$ and $i \in \{0, 1, 2\}$, then either $T(vi) = \emptyset$ or $\{t\} \subseteq T(vi) \subseteq \{t\} \cup \mathbb{P}$; set $X_v = \{i \mid t \in T(vi)\}$;
- if $v \in \{a, b\}^*$, then $T(v) \subseteq X_v \times \mathbb{A} \times X_v$;
- if $v \in \{a, b\}^*$ and $c \in \{a, b\}$, then $X_v \subseteq X_{vc}$ or $X_{vc} \subseteq X_v$;
- if $v \notin \{a, b\}^* \cup \{a, b\}^*\{0, 1, 2\}$, then $T(v) = \emptyset$.

Let T be a valid 2^{P}-labeled A-tree. We now make precise the Kripke structure $K(T)$ over $\mathbb{P}$ and $\mathbb{A}$ whose good tree decomposition is described by T. The structure $K(T)$ should not be confused with T *viewed* as a Kripke structure over P and A (as discussed at the beginning of this section): the original formula φ whose satisfiability is to be decided is interpreted in $K(T)$ whereas the reduction formula $\widehat{\varphi}$ is interpreted in T viewed as a Kripke structure. Define a set of *places* $P = \{u \in \mathsf{A}^* \mid t \in T(u)\}$ and let $\sim$ be the smallest equivalence relation on P which contains all pairs $(vi, vci) \in P \times P$, where $v \in \{a, b\}^*$, $c \in \{a, b\}$, and $0 \le i \le 2$. For $u \in P$, we use $[u]$ to denote the equivalence class of u w.r.t. $\sim$. Now set $K(T) = (X, \{\rightarrow_a \mid a \in \mathbb{A}\}, \rho)$, where:

$$\begin{aligned} X &= \{[u] \mid u \in P\} \\ \rightarrow_a &= \{([vi], [vj]) \mid v \in \{a, b\}^*, (i, a, j) \in T(v)\} \\ \rho([u]) &= \bigcup_{v \in [u]} T(v) \cap \mathbb{P} \end{aligned}$$

The following two lemmas are easily proved.

Lemma 3. *If T is a valid 2^{P}-labeled A-tree, then the Kripke structure $K(T)$ has tree width at most two. Conversely, if K is of tree width at most two, then there exists a valid 2^{P}-labeled A-tree T such that K is isomorphic to $K(T)$.*

Lemma 4. *The set of all valid 2^{P}-labeled A-trees is an ω-regular tree language.*

Now we show how to convert formulas ψ and programs π over $\mathsf{prop}(\varphi)$ and $\mathsf{prog}(\varphi)$ into formulas $\widehat{\psi}$ and programs $\widehat{\pi}$ over P and A such that for every valid 2^{P}-labeled A-tree T we have:

$$\begin{aligned} [\![\widehat{\pi}]\!]_T &\subseteq P \times P \\ \forall u \in P : u \in [\![\widehat{\psi}]\!]_T &\Leftrightarrow [u] \in [\![\psi]\!]_{K(T)} \\ \forall u, v \in P : (u, v) \in [\![\widehat{\pi}]\!]_T &\Leftrightarrow ([u], [v]) \in [\![\pi]\!]_{K(T)} \end{aligned}$$

First we define the auxiliary program

$$\pi_{\sim}^{1} = \bigcup_{i=0}^{2} t? \circ \overline{i} \circ (a \cup b \cup \overline{a} \cup \overline{b}) \circ i \circ t?$$

and let $\pi_{\sim} = (\pi_{\sim}^{1})^{*}$. Note that $[\![\pi_{\sim}]\!]_T$ equals $\sim$. Now, for all $a \in \mathsf{prog}(\varphi)$ and $p \in \mathsf{prop}(\varphi)$ we define

$$\widehat{a} = \bigcup_{i,j \in \{0,1,2\}} \pi_{\sim} \circ \overline{i} \circ (i,a,j)? \circ j \circ \pi_{\sim} \quad \text{and} \quad \widehat{p} = \langle \pi_{\sim} \rangle p.$$

To extend this translation to complex ICPDL formulas and programs, we can simply replace all atomic programs a and formulas p with $\widehat{a}$ and $\widehat{p}$, respectively. From the construction of $\widehat{\varphi}$ and Lemmas 2 and 3 we obtain the following.

Proposition 1. *The formula φ has a model of tree width at most two if and only if there is a valid 2^{P}-labeled A-tree T such that $(T, \varepsilon) \models \langle (0 \cup 1 \cup 2) \circ t? \rangle \widehat{\varphi}$.*

From Theorem 2, Lemma 4, and Proposition 1, we obtain:

Theorem 4. *There is a polynomial time reduction from satisfiability in ICPDL to ω-regular tree satisfiability in ICPDL.*

5 Programs as NFAs over TWAPTAs

Our ultimate goal is to show that ω-regular tree satisfiability in ICPDL can be solved in doubly exponential time. This will be achieved by reducing ω-regular tree satisfiability in ICPDL to the EXP-complete emptiness problem for TWAPTAs. For this we translate ICPDL formulas into TWAPTAs. ICPDL programs will be translated into another special kind of automata, which navigate in a complete Υ-tree by reading symbols from $\Upsilon \cup \overline{\Upsilon}$. Additionally, these automata can make conditional ε-transitions (so called test transitions), which can only be executed if the current tree node is accepted by some TWAPTA. A formal definition follows:

For the rest of this section fix some finite sets $\mathsf{P}, \Gamma = 2^{\mathsf{P}}, \Upsilon$, and a complete Γ-labeled Υ-tree T. In the sequel, we do not require a TWAPTA $\mathcal{T}$ to contain an initial state as a component. For two such TWAPTAs $\mathcal{T}_i = (S_i, \delta_i, \mathrm{Acc}_i)$ $(i \in \{1,2\})$ let $\mathcal{T}_1 \uplus \mathcal{T}_2 = (S_1 \uplus S_2, \delta_1 \uplus \delta_2, \mathrm{Acc}_1 \uplus \mathrm{Acc}_2)$ be their *disjoint union*. Here $\uplus$ denotes the disjoint union, and e.g. $(\mathrm{Acc}_1 \uplus \mathrm{Acc}_2)(s) = \mathrm{Acc}_i(s)$ for the unique i with $s \in S_i$.

A *finite automaton* A over a TWAPTA $\mathcal{T} = (S, \delta, \mathrm{Acc})$ is a tuple $(Q, \rightarrow_A)$, where Q is a finite set of *states*, and $\rightarrow_A$ is a set of transitions of the following kind:

- $q \xrightarrow{a}_A q'$, where $a \in \Upsilon \cup \overline{\Upsilon}$, or
- $q \xrightarrow{\mathcal{T}, s}_A q'$ (these transitions are called *test transitions*),

where $q, q' \in Q, s \in S$. Let $A^{\uparrow} (A^{\downarrow})$ be the automaton that results from A by deleting all $\xrightarrow{a}_A$-transitions ($\xrightarrow{\overline{a}}_A$-transitions), where $a \in \Upsilon$ ($\overline{a} \in \overline{\Upsilon}$). Define the relation $\Rightarrow_A \subseteq (\Upsilon^* \times Q) \times (\Upsilon^* \times Q)$ as the smallest relation such that:

- $(u,p) \Rightarrow_A (ua,q)$ if $p \xrightarrow{a}_A q$ $(a \in \Upsilon)$
- $(ua,p) \Rightarrow_A (u,q)$ if $p \xrightarrow{\overline{a}}_A q$ $(\overline{a} \in \overline{\Upsilon})$
- $(u,p) \Rightarrow_A (u,q)$ if $u \in [\![\mathcal{T}, s]\!]$ and $p \xrightarrow{\mathcal{T},s}_A q$.

For a pair $(p,q) \in Q \times Q$ define $[\![A,p,q]\!]_T = \{(u,v) \in \Upsilon^* \times \Upsilon^* \mid (u,p) \Rightarrow_A^* (v,q)\}$. For each ICPDL formula φ we will construct a TWAPTA $\mathcal{T}$ such that for some state s of $\mathcal{T}$ we have $[\![\mathcal{T}, s]\!]_T = [\![\varphi]\!]_T$. For each ICPDL program π, on the other hand, we will construct a finite automaton A over some TWAPTA $\mathcal{T}$, such that for some states p and q of A we have $[\![A,p,q]\!]_T = [\![\pi]\!]_T$. In the following, the index T will be omitted.

The construction of $\mathcal{T}$ and A above, will be done inductively over the structure of φ and π, respectively. The difficult case is, when π is of the form $\pi_1 \cap \pi_2$. By induction, we have already constructed finite automata A_1 and A_2 over TWAPTAs $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively, such that $[\![A_1,p_1,q_1]\!] = [\![\pi_1]\!]$ and $[\![A_2,p_2,q_2]\!] = [\![\pi_2]\!]$. In order to recognize $[\![A_1,p_1,q_1]\!] \cap [\![A_2,p_2,q_2]\!]$ we would like to make a product construction with A_1 and A_2. But this fails, because a run in T of A_1 and a run in T of A_2, both starting in the tree node u and ending in the tree node v, may completely diverge. In order to avoid this divergence, we next have to normalize finite automata over TWAPTAs, so that loops within a run in T can be shortened by test-transitions. The following construction simplifies the presentation in [8].

Let $\mathcal{T} = (S, \delta, \mathrm{Acc})$ be a TWAPTA and let $A = (Q, \rightarrow_A)$ be a finite automaton over $\mathcal{T}$. Define the relation $\mathrm{loop}_A \subseteq \Upsilon^* \times Q \times Q$ as the smallest set such that:

(i) for all $u \in \Upsilon^*$ and $q \in Q$ we have $(u,q,q) \in \mathrm{loop}_A$,
(ii) if $(ua,p',q') \in \mathrm{loop}_A$, $p \xrightarrow{a}_A p'$ and $q' \xrightarrow{\overline{a}}_A q$, then $(u,p,q) \in \mathrm{loop}_A$,
(iii) if $(u,p',q') \in \mathrm{loop}_A$, $p \xrightarrow{\overline{a}}_A p'$, and $q' \xrightarrow{a}_A q$, then $(ua,p,q) \in \mathrm{loop}_A$,
(iv) if $(u,p,r) \in \mathrm{loop}_A$ and $(u,r,q) \in \mathrm{loop}_A$, then $(u,p,q) \in \mathrm{loop}_A$, and
(v) if $u \in [\![\mathcal{T}, s]\!]$ and $p \xrightarrow{\mathcal{T},s}_A q$ for $s \in S$, then $(u,p,q) \in \mathrm{loop}_A$.

The definition of loop_A allows to prove the following statement by induction over n.

Lemma 5. *We have $(u,p,q) \in \mathrm{loop}_A$ if and only if there exist $n \geq 1, u_1, \ldots, u_n \in \Upsilon^*$, and $q_1, \ldots, q_n \in Q$ such that*

- *$u_1 = u_n = u$,*
- *$q_1 = p, q_n = q$, and*
- *$(u_1,q_1) \Rightarrow_A (u_2,q_2) \Rightarrow_A \cdots \Rightarrow_A (u_n,q_n)$.*

Since the conditions (i)–(v) above can be easily translated into a TWAPTA, we obtain:

Lemma 6. *There is a TWAPTA $\mathcal{U} = (S', \delta', \mathrm{Acc}')$ with $S' = S \uplus (Q \times Q)$ such that*

(i) $[\![\mathcal{U}, s]\!] = [\![\mathcal{T}, s]\!]$ for all $s \in S$,
(ii) $[\![\mathcal{U}, (p,q)]\!] = \{u \in \Upsilon^* \mid (u,p,q) \in \mathrm{loop}_A\}$ for all $(p,q) \in Q \times Q$, and
(iii) $|\mathrm{Acc}'| = |\mathrm{Acc}|$.

Now define a new automaton $B = (Q, \rightarrow_B)$ over the TWAPTA $\mathcal{U}$, that results from A by adding for every pair $(p,q) \in Q \times Q$ the test transition $p \xrightarrow{\mathcal{U},(p,q)}_B q$. For $u,v \in \Upsilon^*$ let $\inf(u,v)$ be the longest common prefix of u and v, it corresponds in the tree T to the lowest common ancestor of u and v.

Lemma 7. *Let $u, v \in \Upsilon^*$ and let $p, q \in Q$. Then the following three statements are equivalent:*

(i) $(u, v) \in [\![A, p, q]\!]$
(ii) $(u, v) \in [\![B, p, q]\!]$
(iii) *there exists* $r \in Q$ *with* $(u, \inf(u, v)) \in [\![B^\uparrow, p, r]\!]$ *and* $(\inf(u, v), v) \in [\![B^\downarrow, r, q]\!]$.

6 ω-regular tree model satisfiability in ICPDL is in 2EXP

In this section, we prove that ω-regular tree satisfiability is in 2EXP. Let $\mathcal{T}_0$ be a TWAPTA over 2^{P}-labeled Υ-trees and let φ be an ICPDL formula. We will translate $\mathcal{T}_0$ and φ into a TWAPTA $\mathcal{T}$ over 2^{P}-labeled Υ-trees such that $L(\mathcal{T}) \neq \emptyset$ if and only if there exists some tree $T \in L(\mathcal{T}_0)$ with $(T, \varepsilon) \models \varphi$. First, we will construct a TWAPTA $\mathcal{T}(\varphi)$ such that for some state s of $\mathcal{T}(\varphi)$ the following equality will hold for all complete 2^{P}-labeled Υ-trees T: $[\![\varphi]\!]_T = [\![\mathcal{T}, s]\!]_T$. The number of states of $\mathcal{T}(\varphi)$ grows exponentially in the size of φ. The size of the priority function will be linear in the size of φ. Our final TWAPTA $\mathcal{T}$ will be the intersection of $\mathcal{T}_0$ and $\mathcal{T}(\varphi)$ (where $\mathcal{T}(\varphi)$ gets the initial state s). Since the time for checking emptiness of $\mathcal{T}$ grows exponentially with the product of the number of states of $\mathcal{T}$ and the size of the priority function of $\mathcal{T}$ (Theorem 3), it follows that that ω-regular tree satisfiability indeed belongs to 2EXP. Together with Theorem 4, this finally proves our main result Theorem 1.

For an ICPDL formula ψ we will inductively construct a TWAPTA $\mathcal{T}(\psi)$ together with a state s of $\mathcal{T}(\psi)$ such that $[\![\psi]\!] = [\![\mathcal{T}(\psi), s]\!]$. For an ICPDL program π, we will inductively construct a TWAPTA $\mathcal{T}(\pi)$ and a finite automaton $A(\pi)$ over $\mathcal{T}(\pi)$ such that $[\![\pi]\!] = [\![A(\pi), p, q]\!]$ for some states p and q of $A(\pi)$.

If $\psi = p$, where $p \in \mathsf{P}$, we put $\mathcal{T}(\psi) = (\{s\}, \delta, s \mapsto 1)$, where for all $Y \subseteq \mathsf{P}$ we have $\delta(s, Y) = \texttt{true}$ if $p \in Y$ and $\delta(s, Y) = \texttt{false}$ otherwise.

If $\psi = \neg\theta$, then $\mathcal{T}(\psi)$ is obtained from $\mathcal{T}(\theta)$ by applying the standard complementation procedure, see e.g. [14], where all positive boolean formulas in the right-hand side of the transition function are dualized and the acceptance condition is complemented by increasing the priority of every state by one.

When ψ is of the form $\langle\pi\rangle\theta$ for a program π and a formula θ, we have inductively already constructed $A = A(\pi)$ with state set Q over a TWAPTA $\mathcal{T}(\pi) = (S_1, \delta_1, \mathrm{Acc}_1)$ such that $[\![\pi]\!] = [\![A, p_0, r_0]\!]$ for some states $p_0, q_0 \in Q$. Too, we have inductively already constructed $\mathcal{T}(\theta) = (S_2, \delta_2, \mathrm{Acc}_2)$ such that $[\![\theta]\!] = [\![\mathcal{T}(\theta), s_2]\!]$ for some state $s_2 \in S_2$. We define the TWAPTA $\mathcal{T}(\psi) = (S, \delta, \mathrm{Acc})$ with $S = Q \uplus S_1 \uplus S_2$. For states in S_1 or in S_2 the transitions of $\mathcal{T}(\psi)$ are as for $\mathcal{T}(\pi)$ or $\mathcal{T}(\theta)$, respectively. For states $q \in Q$ and for $Y \subseteq \mathsf{P}$ we define

$$\begin{aligned}
\delta(q, X) = &\bigvee\{\langle r, \overline{a}\rangle \mid r \in Q, a \in \Upsilon, q \xrightarrow{\overline{a}}_A r\} \ \vee \\
&\bigvee\{\langle r, a\rangle \mid r \in Q, a \in \Upsilon, q \xrightarrow{a}_A r\} \ \vee \\
&\bigvee\{\langle s, \varepsilon\rangle \wedge \langle r, \varepsilon\rangle \mid r \in Q, s \in S_1, q \xrightarrow{\mathcal{T}, s}_A r\} \ \vee \\
&((q = q_0) \wedge \langle s_2, \varepsilon\rangle)
\end{aligned}$$

The priority function Acc is defined by $\mathrm{Acc}(s) = 1$ if $s \in Q$ and $\mathrm{Acc}(s) = (\mathrm{Acc}_1 \uplus \mathrm{Acc}_2)(s)$ for $s \in S_1 \uplus S_2$. We set $\mathrm{Acc}(s) = 1$ for all $s \in Q$ since we want to assure that the automaton $A(\pi)$ is simulated for finitely many steps only, as $\psi = \langle\pi\rangle\theta$ is a diamond formula. We obtain $[\![\psi]\!] = [\![\mathcal{T}(\psi), p_0]\!]$.

Let us now describe the inductive construction of $A(\pi)$ and $\mathcal{T}(\pi)$ for an ICPDL program π.

Case $\pi = a$ for some $a \in \Upsilon \cup \overline{\Upsilon}$. The automaton $A(\pi)$ has two states p and q with the only transition $p \xrightarrow{a} q$. Hence $[\![\pi]\!] = [\![A(\pi), p, q]\!]$.

Case $\pi = \psi?$ We can assume that there exists a TWAPTA $\mathcal{T}(\psi)$ and a state r of $\mathcal{T}(\psi)$ such that $[\![\psi]\!] = [\![\mathcal{T}(\psi), r]\!]$. The TWAPTA $\mathcal{T}(\pi)$ is $\mathcal{T}(\varphi)$. The automaton $A(\pi)$ has two states p and q with the only transition $p \xrightarrow{\mathcal{T}(\pi),r} q$. Hence, we have $[\![\pi]\!] = [\![A(\pi), p, q]\!] = \{(u, u) \mid u \in [\![\mathcal{T}(\psi), r]\!]\}$.

Case $\pi = \pi_1 \cup \pi_2, \pi = \pi_1 \circ \pi_2$*, or* $\pi = \chi^*$ In these cases we construct $A(\pi)$ by using the standard automata constructions for union, concatenation, and Kleene-star. In case $\pi = \pi_1 \cup \pi_2$ or $\pi = \pi_1 \circ \pi_2$ we set $\mathcal{T}(\pi) = \mathcal{T}(\pi_1) \uplus \mathcal{T}(\pi_2)$, whereas for $\pi = \chi^*$ we set $\mathcal{T}(\pi) = \mathcal{T}(\chi)$.

It remains to construct $A(\pi_1 \cap \pi_2)$ and $\mathcal{T}(\pi_1 \cap \pi_2)$. For this, we use the construction of Section 5. Assume that the finite automata $A(\pi_i) = (Q_i, \rightarrow_{A(\pi_i)})$ over the TWAPTA $\mathcal{T}(\pi_i) = (S_i, \delta_i, \mathrm{Acc}_i)$ are already constructed ($i \in \{1, 2\}$). Thus, $[\![A(\pi_i), p_i, q_i]\!] = [\![\pi_i]\!]$ for some states $p_i, q_i \in Q_i$. We first construct the finite automaton $B(\pi_i)$ over the TWAPTA $\mathcal{U}(\pi_i) = (S'_i, \delta'_i, \mathrm{Acc}'_i)$ as described in Section 5. Note that $|S'_i| = |S_i| + |Q_i|^2$. We take $\mathcal{T}(\pi_1 \cap \pi_2) = \mathcal{U}(\pi_1) \uplus \mathcal{U}(\pi_2)$. The finite automaton $A(\pi_1 \cap \pi_2)$ is the product automaton of $B(\pi_1) = (Q_1, \rightarrow_{B(\pi_1)})$ and $B(\pi_2) = (Q_2, \rightarrow_{B(\pi_2)})$, where test transitions can be done asynchronously:

- The state set of $A(\pi_1 \cap \pi_2)$ is $Q_1 \times Q_2$.
- For $a \in \Upsilon \cup \overline{\Upsilon}$ we have $(r_1, r_2) \xrightarrow{a}_{A(\pi_1 \cap \pi_2)} (r'_1, r'_2)$ if and only if $r_1 \xrightarrow{a}_{B(\pi_1)} r'_1$ and $r_2 \xrightarrow{a}_{B(\pi_2)} r'_2$.
- For a state $s \in S'_1 \uplus S'_2$ we have the test transition

$$(r_1, r_2) \xrightarrow{\mathcal{T}(\pi_1 \cap \pi_2), s}_{A(\pi_1 \cap \pi_2)} (r'_1, r'_2)$$

  if and only if either ($s \in S'_1$ and $r_2 = r'_2$ and $r_1 \xrightarrow{\mathcal{U}(\pi_1), s}_{B(\pi_1)} r'_1$) or ($s \in S'_2$ and $r_1 = r'_1$ and $r_2 \xrightarrow{\mathcal{U}(\pi_2), s}_{B(\pi_1)} r'_2$).

Lemma 8. *We have* $[\![A(\pi_1 \cap \pi_2), (p_1, p_2), (q_1, q_2)]\!] = [\![\pi_1 \cap \pi_2]\!]$*. Moreover, if* $\mathcal{T}(\pi_i) = (S_i, \delta_i, \mathrm{Acc}_i)$*,* $A(\pi_i) = (Q_i, \rightarrow_{A(\pi_i)})$*,* $\mathcal{T}(\pi_1 \cap \pi_2) = (S, \delta, \mathrm{Acc})$*, and* $A(\pi_1 \cap \pi_2) = (Q, \rightarrow_{A(\pi_1 \cap \pi_2)})$*, then we have* $|Q| = |Q_1| \cdot |Q_2|$*,* $|S| = |S_1| + |S_2| + |Q_1|^2 + |Q_2|^2$*, and* $|\mathrm{Acc}| = \max\{|\mathrm{Acc}_1|, |\mathrm{Acc}_2|\}$.

A careful analysis of the constructions outlined above, allows us to prove inductively:

Lemma 9. *For every ICPDL formula* ψ *and every ICPDL program* π *we have:*

- *If* $\mathcal{T}(\psi) = (S, \delta, \mathrm{Acc})$ *then* $|S| \leq 2^{|\psi|^2}$ *and* $|\mathrm{Acc}| \leq |\psi|$.

– *If* $A(\pi) = (Q, \rightarrow_{A(\pi)})$ *and* $\mathcal{T}(\pi) = (S, \delta, \mathrm{Acc})$ *then* $|Q| \leq 2^{|\pi|}$, $|S| \leq 2^{|\pi|^2}$, *and* $|\mathrm{Acc}| \leq |\pi|$.

This concludes the proof of our main Theorem 1.

7 Negation of Atomic Programs

We consider extensions of IPDL and ICPDL with negation of programs. It is well known that adding full program negation renders PDL undecidable [9], whereas PDL with program negation restricted to atomic programs remains decidable and EXP - complete [12]. In this section, we show that IPDL and hence also ICPDL become undecidable already when extended with atomic program negation. Since intersection of programs can be defined in terms of program union and (full) program negation, this also yields an alternative proof of the undecidability of PDL with full program negation.

Our proof proceeds by reduction from the undecidable tiling problem of the first quadrant of the plane [3]. A *tiling system* $\mathcal{T} = (T, H, V)$ consists of a finite set of *tile types* T and horizontal and vertical matching relations $H, V \subseteq T \times T$. A *solution* to $\mathcal{T}$ is a mapping $\tau : \mathbb{N} \times \mathbb{N} \rightarrow T$ such that, for all $(x, y) \in \mathbb{N} \times \mathbb{N}$, we have

– if $\tau(x, y) = t$ and $\tau(x + 1, y) = t'$, then $(t, t') \in H$, and
– if $\tau(x, y) = t$ and $\tau(x, y + 1) = t'$, then $(t, t') \in V$.

The tiling problem is to decide, given a tiling system $\mathcal{T}$, whether $\mathcal{T}$ has a solution.

We use IPDL$^{(\neg)}$ to denote the extension of IPDL with negation of atomic programs, which we write as $\neg a$ ($a \in \mathbb{A}$). The semantics of the new constructor is defined in the obvious way, i.e., $[\![\neg a]\!]_K = (X \times X) \setminus [\![a]\!]_K$. To reduce the tiling problem to satisfiability in IPDL$^{(\neg)}$, we give a translation of tiling systems $\mathcal{T} = (T, H, V)$ into formulas $\varphi_{\mathcal{T}}$ of IPDL$^{(\neg)}$ such that $\mathcal{T}$ has a solution if and only if $\varphi_{\mathcal{T}}$ is satisfiable. In the formula $\varphi_{\mathcal{T}}$, we use two atomic programs a_x and a_y for representing the grid $\mathbb{N} \times \mathbb{N}$ and we use the elements of T as atomic propositions for representing tile types. More precisely, $\varphi_{\mathcal{T}}$ is a conjunction consisting of the following conjuncts:

(a) every element of a model of $\varphi_{\mathcal{T}}$ represents an element of $\mathbb{N} \times \mathbb{N}$ and is labelled with a unique tile type:

$$[(a_x \cup a_y)^*]\Big(\bigvee_{t \in T} t \;\wedge \bigwedge_{t,t' \in T, t \neq t'} \neg(t \wedge t')\Big)$$

(b) every element has an a_x-successor and an a_y-successor:

$$[(a_x \cup a_y)^*]\big(\langle a_x \rangle \mathtt{true} \wedge \langle a_y \rangle \mathtt{true}\big)$$

(c) the programs a_x and a_y are confluent:

$$[(a_x \cup a_y)^*]\ [(a_x; a_y) \cap (a_y; \neg a_x)]\mathtt{false}$$

(d) the horizontal and vertical matching conditions are respected:

$$[(a_x \cup a_y)^*]\big(\bigwedge_{t\in T} t \;\Rightarrow\; \big([a_x] \bigvee_{(t,t')\in H} t' \;\wedge\; [a_y] \bigvee_{(t,t')\in V} t'\big)\big).$$

Lemma 10. *$\mathcal{T}$ has a solution if and only if $\varphi_{\mathcal{T}}$ is satisfiable.*

We have thus established the following result.

Theorem 5. *Satisfiability in IPDL$^{(\neg)}$ is undecidable.*

References

1. L. Afanasiev, P. Blackburn, I. Dimitriou, B. Gaiffe, E. Goris, M. J. Marx, and M. de Rijke. PDL for ordered trees. *Journal of Applied Non-Classical Logics*, 15(2), 2005.
2. N. Alechina, S. Demri, and M. de Rijke. A modal perspective on path constraints. *Journal of Logic and Computation*, 13(6):939–956, 2003.
3. R. Berger. The undecidability of the dominoe problem. *Memoirs of the American Mathematical Society*, 66, 1966.
4. R. Danecki. Nondeterministic Propositional Dynamic Logic with intersection is decidable. In *Proc. 5th Symp. Computation Theory*, LNCS 208, pages 34–53, 1984.
5. L. Farinas Del Cerro and E. Orlowska. DAL-a logic for data analysis. *Theoretical Computer Science*, 36(2-3):251–264, 1985.
6. M. J. Fischer and R. E. Ladner. Propositional Dynamic Logic of Regular Programs. *Journal of Computer and System Sciences*, 18(2):194–211, 1979.
7. G. D. Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proc. AAAI94*, pages 205–212, 1994.
8. S. Göller and M. Lohrey. Infinite state model-checking of propositional dynamic logics. In *Proc. CSL 2006*, LNCS 4207, pages 349–364. Springer, 2006.
9. D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. Foundations of computing. The MIT Press, 2000.
10. M. Lange and C. Lutz. 2-ExpTime Lower Bounds for Propositional Dynamic Logics with Intersection. *Journal of Symbolic Logic*, 70(4):1072–1086, 2005.
11. C. Lutz. PDL with intersection and converse is decidable. In *Proc. CSL 2005*, LNCS 3634, pages 413–427. Springer, 2005.
12. C. Lutz and D. Walther. PDL with negation of atomic programs. *Journal of Applied Non-Classical Logics*, 15(2):189–213, 2005.
13. J. Meyer. Dynamic logic for reasoning about actions and agents. In J. Minker, editor, *Logic-Based Artificial Intelligence*, pages 281–311. Kluwer Academic Publishers, 2000.
14. D. Muller and P. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54(2-3):267–276, 1987.
15. V. Pratt. A near-optimal method for reasoning about action. *Journal of Computer and System Sciences*, 20:231–254, 1980.
16. B. ten Cate. The expressivity of XPath with transitive closure. In *Proceedings of the Twenty-Fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 2006)*, pages 328–337. ACM Press, 2006.
17. H. P. van Ditmarsch, W. van der Hoek, and B. P. Kooi. Concurrent dynamic epistemic logic for MAS. In *Proc. AAMAS 2003*, pages 201–208. ACM Press, 2003.
18. M. Y. Vardi. The taming of converse: Reasoning about two-way computations. In *Proc. Logics of Programs*, LNCS 193, pages 413–423. Springer, 1985.
19. M. Y. Vardi. Reasoning about the past with two-way automata. In *Proc. ICALP '98*, LNCS 1443, pages 628–641. Springer, 1998.

A Proof of Lemma 1

Lemma 1. For all $v, u \in V$, $x, y \in X$, $\pi \in \mathsf{subp}(\varphi_0)$, and $\varphi \in \mathsf{subf}(\varphi_0)$,

1. if $t_v = (x, y)$ and v is of type π, then $([v, x], [v, y]) \in [\![\pi]\!]_{K'}$;
2. if $(v, x), (u, y) \in P$ and $([v, x], [u, y]) \in [\![\pi]\!]_{K'}$, then $(x, y) \in [\![\pi]\!]_K$;
3. if $(v, x) \in P$, then $(K, x) \models \varphi$ if and only if $(K', [v, x]) \models \varphi$.

Proof. We prove the three points in the lemma simultaneously by induction on the structure of π and φ. For Point 1, we make a case distinction according to the form of π:

- $\pi \in \mathbb{A}$. Easy by definition of K'.
- $\pi = \overline{a}$ for some $a \in \mathbb{A}$. Easy by definition of K' and the semantics.
- $\pi = \varphi?$. By (†) from Section 3, we have $(x, y) \in [\![\varphi?]\!]_K$ and thus $x = y$ and $(K, x) \models \varphi$. By Point 3 of IH, we get $(K', [v, x]) \models \varphi$. By the semantics and since $x = y$, we obtain $([v, x], [v, y]) \in [\![\varphi?]\!]_{K'}$ as required.
- $\pi = \pi_1 \cap \pi_2$. By construction of T, v has successors u_1 and u_2 of type π_1 and π_2, respectively, such that $t_{u_1} = t_{u_2} = (x, y)$. By Point 1 of IH, we get $([u_i, x], [u_i, y]) \in [\![\pi_i]\!]_{K'}$ for $i \in \{1, 2\}$. By the semantics and since $(u_i, x) \sim (v, x)$ and $(u_i, y) \sim (v, y)$ for $i \in \{1, 2\}$, we obtain $([v, x], [v, y]) \in [\![\pi]\!]_{K'}$.
- $\pi = \pi_1 \cup \pi_2$. Similar to the previous case.
- $\pi = \pi_1 \circ \pi_2$. By construction of T, v has a successor w of type π and such that $t_w = (x, z, y)$ and w has successors u_1 and u_2 of types π_1 and π_2, respectively, such that $t_{u_1} = (x, z)$ and $t_{u_2} = (z, y)$ where $z = C(x, \pi, y)$. By Point 1 of IH, we get $([u_1, x], [u_1, z]) \in [\![\pi_1]\!]_{K'}$ and $([u_2, z], [u_2, y]) \in [\![\pi_2]\!]_{K'}$. It remains to apply the semantics and the fact that $(v, x) \sim (u_1, x)$, $(v, y) \sim (u_2, y)$, and $(u_1, z) \sim (w, z) \sim (u_2, z)$.
- $\pi = \chi^*$. If $x = y$, then $([v, x], [v, y]) \in [\![\pi]\!]_{K'}$. Now assume that $x \neq y$. By construction of T, v has a successor w of type π such that $t_w = (x, z, y)$, and the node w has successors u_1 and u_2 of types χ and π, respectively, such that $t_{u_1} = (x, z)$ and $t_{u_2} = (z, y)$. Here, $z = S(x, \pi, y)$, which means that there exist $n > 0$ and a sequence $x_0, \ldots, x_n \in X$ with
  1. $x_0 = x$ and $x_n = y$;
  2. $(x_i, x_{i+1}) \in [\![\chi]\!]_K$ for all $i < n$;
  3. $x_0, \ldots, x_n$ is a shortest sequence with Properties 1 and 2;
  4. $x_1 = z$.

  By induction on n, we can conclude that $([u_2, z], [u_2, y]) \in [\![\pi]\!]_{K'}$. Moreover, by Point 1 of IH, we get $([u_1, x], [u_1, z]) \in [\![\chi]\!]_{K'}$. It remains to apply the semantics and the fact that $(v, x) \sim (u_1, x)$, $(v, y) \sim (u_2, y)$, and $(u_1, z) \sim (w, z) \sim (u_2, z)$.

For Point 2, we also make a case distinction according to the form of α:

- $\pi = a \in \mathbb{A}$. If $([v, x], [u, y]) \in [\![a]\!]_{K'}$, then at least one of the following holds:
  1. there is $w \in V$ of type a such that $t_w = (x, y)$, $(w, x) \in [v, x]$, and $(w, y) \in [u, y]$;
  2. there is $w \in V$ of type $\overline{a}$ such that $t_w = (y, x)$, $(w, x) \in [v, x]$, and $(w, y) \in [u, y]$.

In Case 1, (†) yields $(x, y) \in [\![a]\!]_K$. Case 2 is analogous.

- $\pi = \overline{a}$. Symmetric to the previous case.
- $\pi = \varphi?$. If $([v, x], [u, y]) \in [\![\varphi?]\!]_{K'}$, then we have $[v, x] = [u, y]$ and $(K', [v, x]) \models \varphi$. By Point 3 of IH, $(K, x) \models \varphi$. By the semantics and since $[v, x] = [u, y]$ implies $x = y$, we get $(x, y) \in [\![\varphi?]\!]_K$.
- The remaining cases are easy using Point 2 of IH and the semantics.

For Point 3, we make a case distinction according to the form of φ.

- If $\varphi \in \mathbb{P}$, then we are done by definition of K';
- The case $\varphi = \neg\psi$ is easy using the semantics and induction hypothesis.
- Let $\varphi = \langle\pi\rangle\psi$. First for the "if" direction. Let $(K', [v, x]) \models \varphi$. Then there is a $(u, y) \in P$ such that $([v, x], [u, y]) \in [\![\pi]\!]_{K'}$ and $(K', [u, y]) \models \psi$. By Point 2 of IH, we get $(x, y) \in [\![\pi]\!]_K$. By Point 3 of IH, we get $(K, y) \models \psi$ and are done by the semantics.
  Now for the "only if" direction. Let $(K, x) \models \varphi$. By construction of T, there is a path $v_0, \ldots, v_n$ in T such that $v = v_0$, v_n is of type singleton, and x belongs to t_{v_i} for all $i \leq n$. Also by construction of T and since $(K, x) \models \langle\pi\rangle\psi$, v_n has a successor u of type π such that $t_u = (x, y)$ for some $y \in X$ such that $y \in [\![\psi]\!]_K$. By Point 1 of IH, u being of type π yields $([u, x], [u, y]) \in [\![\pi]\!]_{K'}$. Moreover, $(v_i, x) \sim (u, x)$ for all $i \leq n$ and thus $([v, x], [u, y]) \in [\![\pi]\!]_{K'}$. By Point 3 of IH, $y \in [\![\psi]\!]_K$ yields $[u, y] \in [\![\psi]\!]_{K'}$ and we are done. □

B Proofs for Section 2

Lemma 2. Every countable Kripke structure of tree width k has a good tree decomposition of width k.

Proof. Let $(T, (X_v)_{v \in V})$ be a tree decomposition for K of width k. In a first step, for every edge (u, v) of T such that neither $X_u \subseteq X_v$ nor $X_v \subseteq X_u$, we add a new node w to the tree T together with the edges (u, w) and (w, v). Of course, the edge (u, v) is deleted. The bag X_w is $X_u \cap X_v$. Now we make T to a rooted tree by choosing an arbitrary root. If a node u of T has $\ell > 2$ many children (possibly $\ell = \aleph_0$), then we can replace this situation by a chain of length ℓ in the usual way (all tree nodes along this chain receive the same bag as u). Finally, we can transform T into a complete binary tree, by just copying bags. □

C Proofs for Section 5

Lemma 5. We have $(u, p, q) \in \mathrm{loop}_A$ if and only if there exist $n \geq 1, u_1, \ldots, u_n \in \Upsilon^*$, and $q_1, \ldots, q_n \in Q$ such that

- $u_1 = u_n = u$,
- $q_1 = p, q_n = q$, and
- $(u_1, q_1) \Rightarrow_A (u_2, q_2) \Rightarrow_A \cdots \Rightarrow_A (u_n, q_n)$.

Proof. Only-if: Assume $(u,p,q) \in \mathrm{loop}_A$. An induction over the shortest proof tree for the fact $(u,p,q) \in \mathrm{loop}_A$ shows that there exist $n \geq 1, u_1, \ldots, u_n \in \Upsilon^*$, and $q_1, \ldots, q_n \in Q$ such that $u_1 = u_n = u$, $q_1 = p$, $q_n = q$ and $(u_1, q_1) \Rightarrow_A (u_2, q_2) \Rightarrow_A \cdots \Rightarrow_A (u_n, q_n)$.
If: Now assume that for some $n \geq 1, u_1, \ldots, u_n \in \Upsilon^*$ and some $q_1, \ldots, q_n \in Q$ we have $u_1 = u_n = u$, $q_1 = p$, $q_n = q$, and $(u_1, q_1) \Rightarrow_A (u_2, q_2) \Rightarrow_A \cdots \Rightarrow_A (u_n, q_n)$. We prove by induction over n, that $(u,p,q) \in \mathrm{loop}_A$ holds. If $n = 1$, then $p = q$ and by rule (i) from the definition of loop_A we have $(u,p,q) \in \mathrm{loop}_A$. If $n = 2$, then there exists a test transition $p \xrightarrow{\mathcal{T},s}_A q$ such that $u \in [\![\mathcal{T}, s]\!]$, hence by rule (v) from the definition of loop_A we have $(u,p,q) \in \mathrm{loop}_A$. Now assume that $n \geq 3$ holds. We distinguish the following cases:

Case 1. There exists $1 < i < n$ such that $u_i = u$. Then, by induction hypothesis, we get $(u, p, q_i), (u, q_i, q) \in \mathrm{loop}_A$. By rule (iv) we get $(u,p,q) \in \mathrm{loop}_A$.

Case 2. There does not exist $1 < i < n$ such that $u_i = u$. Then we distinguish the following cases:

Case 2A. There exists $a \in \Upsilon$ such that $p \xrightarrow{a}_A q_2$, $q_{n-1} \xrightarrow{\overline{a}} q$, and $u_2 = ua = u_{n-1}$. By induction we get $(ua, q_2, q_{n-1}) \in \mathrm{loop}_A$. Thus, by rule (ii) we get $(u,p,q) \in \mathrm{loop}_A$.

Case 2B. There exists $\overline{a} \in \overline{\Upsilon}$ such that $u = va$, $p \xrightarrow{\overline{a}}_A q_2$, $q_{n-1} \xrightarrow{a} q$, and $u_2 = v = u_{n-1}$ for some $v \in \Upsilon^*$. By induction we get $(v, q_2, q_{n-1}) \in \mathrm{loop}_A$. Thus, by rule (iii) we get $(u,p,q) \in \mathrm{loop}_A$. $\square$

Lemma 6. There exists a TWAPTA $\mathcal{U} = (S', \delta', \mathrm{Acc}')$ with state set $S' = S \uplus (Q \times Q)$ such that

(i) for every $s \in S$ we have $[\![\mathcal{U}, s]\!] = [\![\mathcal{T}, s]\!]$,
(ii) for every $(p,q) \in Q \times Q$ we have $[\![\mathcal{U}, (p,q)]\!] = \{u \in \Upsilon^* \mid (u,p,q) \in \mathrm{loop}_A\}$, and
(iii) $|\mathrm{Acc}'| = |\mathrm{Acc}|$.

Proof. For states in S the transitions of $\mathcal{U}$ are the same as for $\mathcal{T}$. For $q \in Q$ and $\gamma \in \Gamma$ we introduce the transition $\delta'((q,q),\gamma) = \texttt{true}$. If $p \neq q$ and $\gamma \in \Gamma$, then we introduce the transition (we write $\bigvee\{\psi_i \mid i \in I\}$ instead of $\bigvee_{i \in I} \psi_i$)

$$\begin{aligned}
\delta'((p,q),\gamma) = {} & \bigvee\{\langle (p',q'), a\rangle \mid p',q' \in Q, a \in \Upsilon, p \xrightarrow{a}_A p', q' \xrightarrow{\overline{a}}_A q\} \ \vee \\
& \bigvee\{\langle (p',q'), \overline{a}\rangle \mid p',q' \in Q, a \in \Upsilon, p \xrightarrow{\overline{a}}_A p', q' \xrightarrow{a}_A q\} \ \vee \\
& \bigvee\{\langle (p,r), \varepsilon\rangle \wedge \langle (r,q), \varepsilon\rangle \mid r \in Q\} \ \vee \\
& \bigvee\{\langle s, \varepsilon\rangle \mid s \in S, p \xrightarrow{\mathcal{T},s}_A q\}.
\end{aligned}$$

We define the priority function Acc' as follows:

$$\mathrm{Acc}'(s') = \begin{cases} \mathrm{Acc}(s) & \text{if } s' \in S \\ 1 & \text{if } s' \in Q \times Q \end{cases}$$

Trivially (iii) holds. We put $\mathrm{Acc}'(p,q) = 1$ for all $p, q \in Q$ since $\mathcal{U}$ should spend only a finite number of steps for verifying whether $(u, p, q) \in \mathrm{loop}_A$. From the definition of δ' it is clear that (i) holds. From the definition of loop_A, the construction of δ' and Acc', and by Lemma 5 it follows that $[\![\mathcal{U}, (p,q)]\!] = \{u \in \Upsilon^* \mid (u, p, q) \in \mathrm{loop}_A\}$, hence (ii) holds. $\square$

Lemma 7. Let $u, v \in \Upsilon^*$ and let $p, q \in Q$. Then the following three statements are equivalent:

(i) $(u, v) \in [\![A, p, q]\!]$
(ii) $(u, v) \in [\![B, p, q]\!]$
(iii) there exists $r \in Q$ with $(u, \inf(u, v)) \in [\![B^{\uparrow}, p, r]\!]$ and $(\inf(u, v), v) \in [\![B^{\downarrow}, r, q]\!]$.

Proof. Trivially (iii) implies (ii). For (ii) implies (i), note that for every test-transition $p' \xrightarrow{\mathcal{U},(p,q)}_B q'$ of B and every $w \in [\![\mathcal{U}, (p', q')]\!]$ we have $(w, p', q') \in \mathrm{loop}_A$, hence $(w, p') \Rightarrow_A^* (w, q')$. Finally, it remains to prove (i) implies (iii). Assume $(u, v) \in [\![A, p, q]\!]$. Then there exist nodes $w_0, \ldots, w_n \in \Upsilon^*$ and states $q_0, \ldots, q_n \in Q$ such that

$$(u, p) = (w_0, q_0) \Rightarrow_A (w_1, q_1) \cdots \Rightarrow_A (w_n, q_n) = (v, q). \tag{1}$$

Let $y_1, \ldots, y_k \in \Upsilon^*$ be the unique nodes such that (a) $y_1 = u$, $y_k = v$, (b) for some $1 \le j \le k$ we have $y_j = \inf(u, v)$, (c) for all $1 \le i < j$ we have $y_i = y_{i+1} a_i$ for some $a_i \in \Upsilon$, and (d) for all $j < i \le k$ we have $y_i = y_{i-1} a_i$ for some $a_i \in \Upsilon$. Define the mapping $\phi : \{1, \ldots, k\} \to \{1, \ldots, n\}$ such that for all $1 \le i \le k$ we have $\phi(i) = \max\{j \mid y_i = w_j\}$. There exist states $q'_1, \ldots q'_k \in Q$ such that the run from equation (1) can be factorized as follows:

$$\begin{gathered}(u, p) = (y_1, q'_1) \Rightarrow_B^* (y_1, q_{\phi(1)}) \Rightarrow_{B^{\uparrow}} (y_2, q'_2) \Rightarrow_B^* (y_2, q_{\phi(2)}) \\ \cdots \Rightarrow_B^* (y_{j-1}, q_{\phi(j-1)}) \Rightarrow_{B^{\uparrow}} (\inf(u, v), q'_j) \Rightarrow_B^* (\inf(u, v), q_{\phi(j)}) \Rightarrow_{B^{\downarrow}} \\ \cdots \Rightarrow_B^* (y_{k-1}, q_{\phi(k-1)}) \Rightarrow_{B^{\downarrow}} (y_k, q'_k) \Rightarrow_B^* (y_k, q_{\phi(k)}) = (v, q)\end{gathered}$$

By the construction of B, every loop $(y_i, q'_i) \Rightarrow_B^* (y_i, q_{\phi(i)})$ can be replaced by a test transition in B (and hence in $B^{\uparrow}$ and $B^{\downarrow}$). Hence, we have $(u, \inf(u, v)) \in [\![B^{\uparrow}, p, q'_j]\!]$ and $(\inf(u, v), v) \in [\![B^{\downarrow}, q'_j, q]\!]$. $\square$

D Proofs for Section 6

Lemma 8. We have $[\![A(\pi_1 \cap \pi_2), (p_1, p_2), (q_1, q_2)]\!] = [\![\pi_1 \cap \pi_2]\!]$. Moreover, if $\mathcal{T}(\pi_i) = (S_i, \delta_i, \mathrm{Acc}_i)$, $A(\pi_i) = (Q_i, \to_{A(\pi_i)})$, $\mathcal{T}(\pi_1 \cap \pi_2) = (S, \delta, \mathrm{Acc})$, and $A(\pi_1 \cap \pi_2) = (Q, \to_{A(\pi_1 \cap \pi_2)})$, then we have $|Q| = |Q_1| \cdot |Q_2|$, $|S| = |S_1| + |S_2| + |Q_1|^2 + |Q_2|^2$, and $|\mathrm{Acc}| = \max\{|\mathrm{Acc}_1|, |\mathrm{Acc}_2|\}$.

Proof. The estimations on the size of Q, S, and Acc are clear by the construction of $\mathcal{T}(\pi_1 \cap \pi_2)$ and $A(\pi_1 \cap \pi_2)$.

For the identity $[\![A(\pi_1\cap\pi_2),(p_1,p_2),(q_1,q_2)]\!] = [\![\pi_1\cap\pi_2]\!]$ note that since $(u,v)\in[\![\pi_1\cap\pi_2]\!]$ if and only if $(u,v)\in[\![\pi_1]\!]$ and $(u,v)\in[\![\pi_2]\!]$, we know by induction that it suffices to prove: $(u,v)\in[\![A(\pi_i),p_i,q_i]\!]$ for all $i\in\{1,2\}$ if and only if $(u,v)\in[\![A(\pi_1\cap\pi_2),(p_1,p_2),(q_1,q_2)]\!]$. So let $(u,v)\in[\![A(\pi_i),p_i,q_i]\!]$ for all $i\in\{1,2\}$. Then Lemma 7 implies the existence of a state $r_i\in Q_i$ such that $(u,\inf(u,v))\in[\![B(\pi_i)^{\uparrow},p_i,r_i]\!]$ and $(\inf(u,v),v)\in[\![B(\pi_i)^{\downarrow},r_i,q_i]\!]$ for all $i\in\{1,2\}$. This implies $(u,\inf(u,v))\in[\![A(\pi_1\cap\pi_2)^{\uparrow},(p_1,p_2),(r_1,r_2)]\!]$ and $(\inf(u,v),v)\in[\![A(\pi_1\cap\pi_2)^{\downarrow},(r_1,r_2),(q_1,q_2)]\!]$. Thus, we have $(u,v)\in[\![A(\pi_1\cap\pi_2),(p_1,p_2),(q_1,q_2)]\!]$. On the other hand, any run witnessing $(u,v)\in[\![A(\pi_1\cap\pi_2),(p_1,p_2),(q_1,q_2)]\!]$ is a witness for $(u,v)\in[\![B(\pi_i),p_i,q_i]\!]$ for all $i\in\{1,2\}$. By Lemma 7 we obtain $(u,v)\in[\![A(\pi_i),p_i,q_i]\!]$ for all $i\in\{1,2\}$. $\square$

Lemma 9. For every ICPDL formula ψ and every ICPDL program π we have:

- If $\mathcal{T}(\psi)=(S,\delta,\mathrm{Acc})$ then $|S|\leq 2^{|\psi|^2}$ and $|\mathrm{Acc}|\leq|\psi|$.
- If $A(\pi)=(Q,\rightarrow_{A(\pi)})$ and $\mathcal{T}(\pi)=(S,\delta,\mathrm{Acc})$ then $|Q|\leq 2^{|\pi|}$, $|S|\leq 2^{|\pi|^2}$, and $|\mathrm{Acc}|\leq|\pi|$.

Proof. We prove the lemma via mutual induction over the structure of ψ and π.

Base. If $\psi=p\in\mathbb{P}$, then $\mathcal{T}(\psi)$ has $1\leq 2^{|\psi|}$ states and the size of the priority function of $\mathcal{T}(\psi)$ is $1=|\psi|$.

Assume $\pi=a\in\Upsilon\cup\overline{\Upsilon}$. By construction, the automaton $A(\pi)=(Q,\rightarrow_{A(\pi)})$ does not have any transitions over some TWAPTA. Moreover, we have $|Q|=2=2^{|\pi|}$.

Inductive step. In case $\psi=\neg\theta$ and $\mathcal{T}(\theta)=(S',\delta',\mathrm{Acc}')$, then by the standard complementation of $\mathcal{T}(\theta)$ yielding $\mathcal{T}(\psi)$ we have $S=S'$. Moreover, by induction we have $|\mathrm{Acc}'|\leq|\theta|$, hence $|\mathrm{Acc}|=|\mathrm{Acc}'|+1\leq|\theta|+1=|\psi|$.

Now assume $\psi=\langle\pi\rangle\theta$. Let $A(\pi)=(Q,\rightarrow_{A(\pi)})$, $\mathcal{T}(\pi)=(S_1,\delta_1,\mathrm{Acc}_1)$, and $\mathcal{T}(\theta)=(S_2,\delta_2,\mathrm{Acc}_2)$. Then by construction we have:

$$\begin{aligned}
|S| &= |Q|+|S_1|+|S_2| \\
&\overset{\text{induction}}{\leq} 2^{|\pi|}+2^{|\pi|^2}+2^{|\theta|} \\
&\leq 2^{|\pi|+|\pi|^2+|\theta|} \\
&\leq 2^{|\pi|^2+2\cdot|\pi|\cdot|\theta|+|\theta|^2} \\
&= 2^{(|\pi|+|\theta|)2} \\
&= 2^{|\psi|^2}
\end{aligned}$$

The cases $\pi=\chi^*$ and $\pi=\psi?$ are easy to analyze. Next, assume $\pi=\pi_1\cup\pi_2$ or $\pi=\pi_1\circ\pi_2$ and let Q_i be the state set of $A(\pi_i)$ and $\mathcal{T}(\pi_i)=(S_i,\delta_i,\mathrm{Acc}_i)$ $(i\in\{1,2\})$. By the standard construction we get:

$$\begin{aligned}
|Q| &= |Q_1|+|Q_2| \\
&\overset{\text{induction}}{\leq} 2^{|\pi_1|}+2^{|\pi_2|} \\
&\leq 2^{|\pi_1|+|\pi_2|} \\
&= 2^{|\pi|}
\end{aligned}$$

The estimation of $|S|$ and $|\mathrm{Acc}|$ is straightforward again.

Now assume that $\pi = \pi_1 \cap \pi_2$. Let Q_i be the state set of $A(\pi_i)$ and $\mathcal{T}(\pi_i) = (S_i, \delta_i, \mathrm{Acc}_i)$ $(i \in \{1,2\})$. By Lemma 8 we have $|Q| = |Q_1| \cdot |Q_2|$ and $|S| = |S_1| + |S_2| + |Q_1|^2 + |Q_2|^2$. Hence, we get:

$$\begin{aligned}
|Q| &= |Q_1| \cdot |Q_2| \\
&\overset{\text{induction}}{\leq} 2^{|\pi_1|} \cdot 2^{|\pi_2|} \\
&= 2^{|\pi_1|+|\pi_2|} \\
&= 2^{|\pi|} \\
|S| &= |S_1| + |S_2| + |Q_1|^2 + |Q_2|^2 \\
&\overset{\text{induction}}{\leq} 2^{|\pi_1|^2} + 2^{|\pi_2|^2} + 2^{2\cdot|\pi_1|} + 2^{2\cdot|\pi_2|} \\
&\leq 2^{(|\pi_1|+|\pi_2|)^2} \\
&= 2^{|\pi|^2}
\end{aligned}$$

By induction we get $|\mathrm{Acc}_i| \leq |\pi_i|$ for $i \in \{1,2\}$. Thus, $\mathrm{Acc} = \max\{|\mathrm{Acc}_1|, |\mathrm{Acc}_2|\} \leq |\mathrm{Acc}_1| + |\mathrm{Acc}_2| \leq |\pi_1| + |\pi_2| = |\pi|$. $\square$

E Proof of Lemma 10

Lemma 10. $\mathcal{T}$ has a solution if and only if $\varphi_{\mathcal{T}}$ is satisfiable.

Proof. Since the "$\Rightarrow$" direction is simple, we only prove the "$\Leftarrow$" direction. Thus, let $\varphi_{\mathcal{T}}$ be satisfiable and $K = (X, \{\rightarrow_{a_x}, \rightarrow_{a_y}\}, \rho)$ a model of $\varphi_{\mathcal{T}}$. We have to construct a solution τ to $\mathcal{T}$. To prepare for this, we first define a mapping $\pi : \mathbb{N} \times \mathbb{N} \rightarrow X$, which is done in two steps. In the first step, we pick $x \in [\![\varphi_{\mathcal{T}}]\!]_K$ and set $\pi(i,j) = x$. Next, we define $\pi(i,0)$ for all $i > 0$ as follows: Assume that $\pi(i-1,0)$ is already defined. By (b) from Section 7, we can choose $x \in X$ such that $(\pi(i-1,0), x) \in [\![a_x]\!]_K$ and set $\pi(i,0) = x$. Finally, we define $\pi(i,j)$ for all $j > 0$: Assume that $\pi(i,j-1)$ is already defined. By (b) from Section 7, we can choose $y \in X$ such that $(\pi(i,j-1), y) \in [\![a_y]\!]_K$ and set $\pi(i,j) = y$. By construction, we have $(\pi(i,j-1), \pi(i,j)) \in [\![a_y]\!]_K$ for every $i \geq 0$ and $j > 0$. Moreover, the confluence property (c) from Section 7 implies that also $(\pi(i-1,j), \pi(i,j)) \in [\![a_x]\!]_K$ for every $i > 0$ and $j \geq 0$.

The resulting mapping π gives rise to a mapping $\tau : \mathbb{N} \times \mathbb{N} \rightarrow T$ in the obvious way: by (a), we can define $\tau(i,j)$ as the unique $t \in T$ with $\pi(i,j) \in [\![t]\!]_K$. Finally, by (d), τ is a solution to $\mathcal{T}$. $\square$