PDL with Intersection and Converse is 2EXP-complete

Stefan Göller * **

Institut für Informatik, Universität Leipzig, Germany
goeller@informatik.uni-leipzig.de

Abstract. The logic ICPDL is the expressive extension of Propositional Dynamic Logic (PDL), which admits intersection and converse as program operators. The result of this paper is containment of ICPDL-satisfiability 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 (IPDL). The proof proceeds showing that every satisfiable ICPDL formula has model of tree width at most two. Next, we reduce satisfiability in ICPDL to ω-regular tree satisfiability in ICPDL. In the latter problem the set of possible models is restricted to trees of an ω-regular tree language. In the final step, ω-regular tree satisfiability is reduced the emptiness problem for alternating two-way automata on infinite trees. In this way, a more elegant proof is obtained for Danecki's difficult result that satisfiability in IPDL 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 [7]. In PDL, there are two syntactic entities: formulas, built from boolean and modal operators and interpreted as sets of worlds 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, many of them allow additional operators on programs. Three prominent such 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 [11] and references therein. While some of these extensions such as CPDL are well-suited for reasoning about programs, many of them aim at the numerous other applications that PDL has found since its invention. Notable examples of such applications include agent-based systems [14], regular path constraints [2], and XML-querying [1, 17, 18]. In artifical intelligence, PDL received attention due to its close relationship to description logics [9] and epistemic logic [20, 19].

* This article is joint work with Markus Lohrey and Carsten Lutz and based on the paper [10] which appeared at FoSSaCS 2007.

** The author is supported by the DFG project GELO.

Dagstuhl Seminar Proceedings 07441
Algorithmic-Logical Theory of Infinite Structures
http://drops.dagstuhl.de/opus/volltexte/2008/1409

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 [7, 16]. The EXP upper bound can be extended to CPDL and can even be established for several extensions thereof [21]. In contrast, the precise complexity of satisfiability for IPDL was a long 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 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 Lutz [12]. Regarding NPDL, it is long known that satisfiability is undecidable [11]. As recently shown in [11], the fragment of NPDL in which program negation is restricted to atomic programs is decidable and EXP-complete.

In this paper, we prove that the complexity of satisfiability in ICPDL, the extension of PDL with both converse and intersection, is complete for deterministic doubly exponential time. Decidability was shown by Lutz in [13] 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) [6] is a fragment of ICPDL (but not of IPDL) and thus inherits the 2EXP upper bound; so far only decidability of satisfiability of DAL was known. And third, our result has applications in description logic and epistemic logic, see [13] for more details.

Our main result is proved in three clearly separated parts. In part one (Section 3), we establish a 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 (recall that trees have tree width one). This part of the proof of our main result is comparable to Danecki's result that every satisfiable IPDL formula has a special model which can be encoded by a tree.

In part two of our proof (Section 4), 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 with a parity acceptance condition that runs on infinite node-labeled trees and can move upwards and downwards in the tree. Infinite node-labeled

trees can be viewed in a natural way as Kripke structures and thus we can interpret 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 of φ? Our reduction of satisfiability in ICPDL to this problem is based on a suitable encoding of all countable Kripke structures of tree width at most two. The TWAPTA constructed in the reduction accepts precisely such encodings.

Finally, in part three (Section 5) we reduce ω-regular tree satisfiability in ICPDL to the non-emptiness problem for TWAPTAs. The latter problem was shown to be EXP-complete in [22]. Since our reduction of ω-regular tree satisfiability in ICPDL to TWAPTA-non-emptiness involves an exponential blow-up in automata size, we obtain a 2EXP upper bound for ω-regular tree satisfiability in ICPDL and also for standard satisfiability in ICPDL.

As a corollary of our proof, we obtain that satisfiability in the extension of PDL with converse and the *loop-construct* (briefly, loop-CPDL) belongs to EXP. The loop-construct allows to express that execution of program π allows to return to the initial world. Membership of satisfiability in PDL with loop in EXP was shown by Danecki [5].

2 ICPDL basics

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

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

The *size* of ICPDL formulas and programs (denoted $|\cdot|$) is defined by mutual induction: $|p| = |a| = |\overline{a}| = 1$ for all $p \in \mathbb{P}$ and $a \in \mathbb{A}$, $|\neg\psi| = |\psi?| = |\psi| + 1$, $|\langle\pi\rangle\psi| = |\pi| + |\psi| + 1$, $|\pi_1 \text{ op } \pi_2| = |\pi_1| + |\pi_2| + 1$ for op $\in \{\cup, \cap, \circ\}$, and $|\pi^*| = |\pi| + 1$. The semantics of ICPDL is defined in terms of Kripke structures. A *Kripke structure* is a tuple $K = (X, \{\rightarrow_a \mid a \in \mathbb{A}\}, \{X_p \mid p \in \mathbb{P}\})$, where

- X is a set of *worlds*,
- $\rightarrow_a \subseteq X \times X$ is a *transition relation* for each $a \in \mathbb{A}$, and
- $X_p \subseteq X$ is a unary relation for each $p \in \mathbb{P}$.

Given a Kripke structure $K = (X, \{\rightarrow_a \mid a \in \mathbb{A}\}, \{X_p \mid p \in \mathbb{P}\})$, we define for each ICPDL program π a binary relation $[\![\pi]\!]_K \subseteq X \times X$ and for each ICPDL

formula φ a subset $[\![\varphi]\!]_K \subseteq X$, using mutual induction as follows: [1]

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

Note that applying the converse operator $^-$ to atomic programs is not a restriction since it commutes with all other operators. Also note that $\varphi_1 \wedge \varphi_2$ can be written as $\langle\varphi_1?\rangle\varphi_2$ and $\texttt{true} = \neg(p \wedge \neg p)$ for some $p \in \mathbb{P}$. If $x \in [\![\varphi]\!]_K$ for some $x \in X$, then the Kripke structure K is a *model* of φ. The formula φ is *satisfiable* if there exists a model of φ. The *satisfiability problem* in ICPDL is to decide, given an ICPDL formula φ, whether φ is satisfiable.

3 Models of tree width two suffice

The goal of this section is to introduce an operator $\oplus$ that accomplishes the following: Applying $\oplus$ to a Kripke structure K together with an initial world x_0 of K yields a Kripke structure $K^\oplus$ of tree width (to be defined below) at most two together with an initial world $x_0^\oplus$ of $K^\oplus$ such that for every ICPDL formula φ we have $x_0 \in [\![\varphi]\!]_K$ if and only if $x_0^\oplus \in [\![\varphi]\!]_{K^\oplus}$.

We start with defining tree decompositions and the tree width of Kripke structures. Let

$$K = (X, \{\rightarrow_a \mid a \in \mathbb{A}\}, \{X_p \mid p \in \mathbb{P}\})$$

be a Kripke structure. A *tree decomposition* of K is a tuple $(T, (B_v)_{v \in V})$, where $T = (V, E)$ is an undirected tree, B_v is a subset of X (called a *bag*) for all $v \in V$, and the following conditions are satisfied:

1. $\bigcup_{v \in V} B_v = X$,
2. For every transition $x \rightarrow_a y$ of K, there exists $v \in V$ with $x, y \in B_v$, and
3. For every $x \in X$, the set $\{v \in V \mid x \in B_v\}$ is a connected subset in T.

The tree decomposition $(T, (B_v)_{v \in V})$ is countable, if V is countable. The width of the tree decomposition is the supremum of $\{|B_v| - 1 \mid v \in V\}$. The *tree width* of a Kripke structure K is the minimal k such that K has a tree decomposition of width k. We have the following result.

[1] We overload notation and use e.g. $\circ$ both as a program operator of ICPDL and to denote the composition operator for binary relations, i.e., $R \circ S = \{(a,b) \mid \exists c : (a,c) \in R, (c,b) \in S\}$.

Theorem 1. *Let K be a Kripke structure and let x_0 be a world of K. Then there exists a Kripke structure $K^{\oplus}$ of tree width at most two and a world $x_0^{\oplus}$ of $K^{\oplus}$ such that for every ICPDL formula φ we have $x_0 \in [\![\varphi]\!]_K$ if and only if $x_0^{\oplus} \in [\![\varphi]\!]_{K^{\oplus}}$. Moreover, if K is countable then $K^{\oplus}$ has a countable tree decomposition of width at most two.*

Let us outline the construction the proof of Theorem 1. Fix a Kripke structure $K = (X, \{\rightarrow_a \mid a \in \mathbb{A}\}, \{X_p \mid p \in \mathbb{P}\})$ and a world $x_0 \in X$ of K. Firstly, we inductively define an undirected tree $T = (V, E)$ together with a node labeling $t_v \in X \cup X^2 \cup X^3$ for each $v \in V$ given by the following rules:

1. Start the construction with a root $v_0 \in V$ and put $t_{v_0} = x_0$.
2. If $v \in V$ and $t_v = x \in X$, then for every $y \in X$ add a child w of v and set $t_w = (x, y)$.
3. If $v \in V$ and $t_v = (x, y)$, then add
   - for every $z \in X$ a child w of v with $t_w = (x, z, y)$ and
   - a child w' with $t_{w'} = y$.
4. If $v \in V$ and $t_v = (x, z, y)$, then add children w_1 and w_2 with $t_{w_1} = (x, z)$ and $t_{w_2} = (z, y)$.

We assume that successors are added at most once to each node and that the rules are applied in a breadth first manner. A *place* is a pair $(v, x) \in V \times X$ such that t_v contains x. We denote by P_T the set of places of T and define $\approx$ to be the smallest equivalence relation on P_T that contains all pairs of the form $((v, x), (u, x))$ such that $\{v, u\} \in E$. Whenever $(v, x) \in P_T$ we denote by $[v, x]$ the equivalence class of (v, x) with respect to $\approx$. Finally, let us define $K^{\oplus} = (X', \{\rightarrow'_a \mid a \in \mathbb{A}\}, \{X'_p \mid p \in \mathbb{P}\})$ as follows,

- $X' = \{[v, x] \mid (v, x) \in P_T\}$,
- $[v, x] \rightarrow'_a [v', y]$ whenever there exists some $u \in V$ such that $(v, x) \approx (u, x)$,$(v', y) \approx (u, y)$ and $x \rightarrow_a y$, and
- $X'_p = \{[v, x] \in X' \mid x \in X_p\}$.

We put $x_0^{\oplus} = [v_0, x_0]$. Theorem 1 follows from Point (3) of the following lemma.

Lemma 1. *For all $(v, x), (u, y) \in P_T$, all programs π and all formulas φ:*

(1) if $t_v = (x, y)$ and $(x, y) \in [\![\pi]\!]_K$, then $([v, x], [v, y]) \in [\![\pi]\!]_{K^{\oplus}}$;
(2) if $([v, x], [u, y]) \in [\![\pi]\!]_{K^{\oplus}}$, then $(x, y) \in [\![\pi]\!]_K$;
(3) $x \in [\![\varphi]\!]_K$ if and only if $[v, x] \in [\![\varphi]\!]_{K^{\oplus}}$.

4 Reduction from satisfiability to ω-regular tree satisfiability

Informally, the ω-regular tree satisfiability problem in ICPDL is to decide, given an ICPDL formula φ and a two-way alternating parity tree automaton (TWAPTA — the definition will be given below) $\mathcal{T}$, whether there is an infinite tree in $L(\mathcal{T})$ that, when viewed as a Kripke structure, is a model of φ.

Let Σ_N be a finite node alphabet and Σ_E a finite edge alphabet. A *Σ_N-labeled Σ_E-tree* is a partial function $T : \Sigma_E^* \to \Sigma_N$ such that the set of *nodes* $\mathrm{dom}(T)$ is prefix-closed. If $\mathrm{dom}(T) = \Sigma_E^*$, then T is called *complete*. In the rest of the paper we mostly work with complete trees. For a node $va \in \mathrm{dom}(T)$ with $a \in \Sigma_E$, we say that the node va is the *(a-)successor* of v and v is the *(a-)predecessor* of va. We use $\mathrm{tree}(\Sigma_N, \Sigma_E)$ to denote the set of all complete Σ_N-labeled Σ_E-trees. If Σ_E is not important, we simply talk of Σ_N-labeled trees.

The trees accepted by TWAPTAs are complete 2^{P}-labeled A-trees, where $\mathsf{A} \subseteq \mathbb{A}$ and $\mathsf{P} \subseteq \mathbb{P}$ are finite sets of atomic propositions and atomic programs, respectively. Such a tree T can be identified with the Kripke structure $(\mathsf{A}^*, \{\to_a | a \in \mathsf{A}\}, \{T_p \mid p \in \mathsf{P}\})$, where $\to_a = \{(u, ua) \mid u \in \mathsf{A}^*\}$ for all $a \in \mathsf{A}$ and $T_p = \{u \in \mathsf{A}^* \mid p \in T(u)\}$ for $p \in \mathsf{P}$. Observe that Kripke structures derived in this way are total w.r.t. A and deterministic, i.e., the transition relation $\to_a$ is a total function for all $a \in \mathsf{A}$.

For a finite set X, we denote by $\mathcal{B}^+(X)$ the set of all *positive boolean formulas* where the elements of X are used as variables. The constants `true` and `false` are admitted, i.e. we have $\mathtt{true}, \mathtt{false} \in \mathcal{B}^+(X)$ for any set X. A subset $Y \subseteq X$ can be seen as a valuation in the obvious way, i.e., all elements of Y are assigned true and all elements of $X \setminus Y$ are assigned false. For an edge alphabet Σ_E, let $\overline{\Sigma_E} = \{\overline{a} \mid a \in \Sigma_E\}$ be a disjoint copy of Σ_E. For $u \in \Sigma_E^*$ and $d \in \Sigma_E \cup \overline{\Sigma_E} \cup \{\varepsilon\}$ define

$$u \cdot d = \begin{cases} ud & \text{if } d \in \Sigma_E \\ u & \text{if } d = \varepsilon \\ v & \text{if there exists } a \in \Sigma_E \text{ with } d = \overline{a} \text{ and } u = va \\ \text{undefined} & \text{if } d = \overline{a} \text{ for } a \in \Sigma_E \text{ but } u \text{ does not end with } a \end{cases}$$

A *two-way alternating parity tree automaton (TWAPTA)* over complete Σ_N-labeled Σ_E-trees is a tuple $\mathcal{T} = (S, \delta, s_0, \mathrm{Acc})$, where

- S is a finite non-empty set of *states*,
- $\delta : S \times \Sigma_N \to \mathcal{B}^+(\mathrm{ext}(\Sigma_E))$ is the *transition function*, where $\mathrm{ext}(\Sigma_E) := S \times (\Sigma_E \cup \overline{\Sigma_E} \cup \{\varepsilon\})$ is the set of *moves*,
- $s_0 \in S$ is the *initial state*, and
- $\mathrm{Acc} : S \to \mathbb{N}$ is the *priority function*.

For $s \in S$ and $d \in \Sigma_E \cup \overline{\Sigma_E} \cup \{\varepsilon\}$ we write the corresponding move as $\langle s, d \rangle$. Intuitively, a move $\langle s, a \rangle$, with $a \in \Sigma_E$, means that the automaton sends a copy of itself in state s to the a-successor of the current tree node. Similarly, $\langle s, \overline{a} \rangle$ means to send a copy to the a-predecessor (if existing), and $\langle s, \varepsilon \rangle$ means to stay in the current node. Formally, the behaviour of TWAPTAs is defined in terms of runs. Let $\mathcal{T}$ be a TWAPTA as above, $T \in \mathrm{tree}(\Sigma_N, \Sigma_E)$, $u \in \Sigma_E^*$ a node, and $s \in S$ a state of $\mathcal{T}$. An *(s, u)-run* of $\mathcal{T}$ on T is a (not necessarily complete) $(S \times \Sigma_E^*)$-labeled tree T_R such that

- $T_R(\varepsilon) = (s, u)$, and

– for all $\alpha \in \mathrm{dom}(T_R)$, if $T_R(\alpha) = (p, v)$ and $\delta(p, T(v)) = \theta$, then there is a subset $Y \subseteq \mathrm{ext}(\Sigma_E)$ that satisfies θ and such that for all $(p', d) \in Y$, $v \cdot d$ is defined and there exists a successor β of α in T_R with $T_R(\beta) = (p', v \cdot d)$.

We say that an (s, u)-run T_R is *successful* if for every infinite path $\alpha_1\alpha_2\cdots$ in T_R (which is assumed to start at the root), the number

$$\min\{\mathrm{Acc}(s) \mid s \in S \text{ with } T_R(\alpha_i) \in \{s\} \times \Sigma_E^* \text{ for infinitely many } i\}$$

is even. For $s \in S$ define

$$\begin{aligned}[\![\mathcal{T}, s]\!] = \{(T, u) \mid\ & T \in \mathrm{tree}(\Sigma_N, \Sigma_E),\ u \in \Sigma_E^*, \text{ and} \\ & \text{there exists a successful } (s, u)\text{-run of } \mathcal{T} \text{ on } T\} \text{ and} \\ [\![\mathcal{T}]\!] = [\![\mathcal{T}, s_0]\!].\end{aligned}$$

Now the language $L(\mathcal{T})$ accepted by $\mathcal{T}$ is defined as the set of all $T \in \mathrm{tree}(\Sigma_N, \Sigma_E)$ such that $(T, \varepsilon) \in [\![\mathcal{T}]\!]$. We remark that our model of TWAPTAs differs slightly from other definitions that can be found in the literature. First, we run TWAPTAs only on complete trees, which will facilitate some technical constructions later on. Second, standard TWAPTAs have moves of the form $(s, -1)$ for moving to the parent node. In our model, we use moves of the form $(s, \overline{a})$, which can only be executed if the current node is an a-successor of its parent node. It is not hard to see that these two models are equivalent. In particular, it is easy to see that the following result of Vardi also applies to our version of TWAPTAs. For a TWAPTA $\mathcal{T} = (S, \delta, s_0, \mathrm{Acc})$, we define its *size* $|\mathcal{T}| := |S|$ as its number of states and we define its *index* $i(\mathcal{T})$ as $\max\{\mathrm{Acc}(s) \mid s \in S\}$. The size $|\delta|$ of the transition function δ is the sum of the sizes of all positive Boolean functions that appear in the range of δ.

Theorem 2 ([22]). *For a given TWAPTA $\mathcal{T}$ with transition function δ, it can be checked in time $2^{(|\mathcal{T}|+|i(\mathcal{T})|)^{\mathcal{O}(1)}} \cdot |\delta|^{\mathcal{O}(1)}$ whether $L(\mathcal{T}) \neq \emptyset$.*

We can now formally define ω-regular tree satisfiability. Let φ be an ICPDL formula, let $\mathsf{A} = \{a \in \mathbb{A} \mid a \text{ occurs in } \varphi\}$ and $\mathsf{P} = \{p \in \mathbb{P} \mid p \text{ occurs in } \varphi\}$. The formula φ is *satisfiable* with respect to a TWAPTA $\mathcal{T} = (S, \delta, s_0, \mathrm{Acc})$ over 2^{P}-labeled A-trees, if there is a $T \in L(\mathcal{T})$ such that $\varepsilon \in [\![\varphi]\!]_T$. Thus, *$\omega$-regular tree satisfiability* is the problem to decide, given such φ and $\mathcal{T}$, whether φ is satisfiable with respect to $\mathcal{T}$.

4.1 The reduction

Before we reduce satisfiability in ICPDL to ω-regular tree satisfiability, we use a result of Flum.

Theorem 3 ([8]). *Every satisfiable formula of least fixed point logic has a countable model.*

Since ICPDL is a fragment of least fixed point logic, we obtain the following corollary.

Corollary 1. *Every satisfiable ICPDL has a countable model.*

Next, we use good tree decompositions. A tree decomposition $(U, (B_v)_{v \in V})$ is *good* if $U = (V = \{a, b\}^*, E = \{\{v, vc\} \mid v \in \{a, b\}^*, c \in \{a, b\}\})$ is the complete binary tree and $\{u, v\} \in E$ imples $B_v \subseteq B_u$ or $B_u \subseteq B_v$. Clearly, every good tree decomposition is countable. The following lemma is not hard to show.

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

Thus, by Theorem 1, Corollary 1 and Lemma 2 every satisfiable ICPDL formula has a model that has a good tree decomposition of width at most two.

To reduce satisfiability in ICPDL to ω-regular tree satisfiability, we show how to translate an ICPDL formula φ into an ICPDL formula φ' and a TWAPTA $\mathcal{T}$ such that φ is satisfiable if and only if φ' is satisfiable with respect to $\mathcal{T}$. The formula φ' uses atomic propositions and atomic programs

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

where $\mathrm{prop}(\varphi) = \{p \in \mathbb{P} \mid p \text{ occurs in } \varphi\}$ and $\mathrm{prog}(\varphi) = \{a \in \mathbb{A} \mid a \text{ occurs in } \varphi\}$. The TWAPTA $\mathcal{T}$ works on 2^{P}-labeled A-trees. Intuitively, each tree T accepted by $\mathcal{T}$ encodes a Kripke structure K together with a good tree decomposition of K of width at most two, and T is a model of φ' if and only if K is a model of φ.

We first describe the mentioned encoding on an intuitive level. Let K be a Kripke structure and $(U, (B_v)_{v \in V})$ a good tree decomposition of K of width at most two, with $U = (V, E)$, i.e. $V = \{a, b\}^*$, with ε the root of the tree U and for each $v \in V$, va and vb the children of v. The tree T has roughly the structure of U. In particular, each node $v \in U$ is described by the same node v in T, together with additional children $v0$, $v1$, and $v2$ that v has in T. Intuitively, we think of each node v in U as providing three slots which can be empty or filled with a world of the Kripke structure K, i.e. the three slots correspond to an ordered representation of B_v. The additional successors $v0, v1, v2$ of v in T describe these three slots. This explains our choice of A. When slot vi ($i \in \{0, 1, 2\}$) is occupied by a world of K, then vi receives the special label $t \in \mathsf{P}$ in T. Additionally, each node vi is labeled with the same atomic propositions as the world in K that it represents (if any). Information about the transitions of K are stored in T using nodes from $\{a, b\}^*$. For example, if there is a γ-transition in K from the world represented by vi to the world represented by vj, then the labeling of v contains the tuple (i, γ, j). We now formally define the described encoding. 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 $B_v := \{i \mid t \in T(vi)\}$;
- if $v \in \{a, b\}^*$, then $T(v) \subseteq B_v \times \mathrm{prog}(\varphi) \times B_v$;
- if $v \in \{a, b\}^*$ and $c \in \{a, b\}$, then $B_v \subseteq B_{vc}$ or $B_{vc} \subseteq B_v$;
- 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 $\mathrm{prop}(\varphi)$ and $\mathrm{prog}(\varphi)$ that is described by T. Define a set of *places*

$$P_T = \{u \in \{a,b\}^*\{0,1,2\} \mid t \in T(u)\}$$

and let $\sim$ be the smallest equivalence relation on P_T which contains all pairs $(vi, vci) \in P_T \times P_T$, where $v \in \{a,b\}^*$, $c \in \{a,b\}$, and $0 \leq i \leq 2$. For $u \in P_T$, we use $[u]$ to denote the equivalence class of u with respect to $\sim$. Now set

$$K(T) = (X, \{\rightarrow_\gamma | \ \gamma \in \mathrm{prog}(\varphi)\}, \{X_p \mid p \in \mathrm{prop}(\varphi)\}),$$

where

$$\begin{array}{rcl} X & = & \{[u] \mid u \in P_T\}, \\ \rightarrow_\gamma & = & \{([vi],[vj]) \mid v \in \{a,b\}^*, (i,\gamma,j) \in T(v)\}, \text{ and} \\ X_p & = & \{[u] \in X \mid p \in T(u)\}. \end{array}$$

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 Section 4: the original formula φ whose satisfiability is to be decided is interpreted in $K(T)$ whereas the reduction formula φ', to be defined below, is interpreted in T viewed as a Kripke structure over P and A. The following lemma, which is easy to prove, establishes the correctness of our encoding.

Lemma 3. *If K is a Kripke structure that has a good tree decomposition of width two, then there exists a valid $T \in$ tree$(2^{\mathsf{P}}, \mathsf{A})$ such that K is isomorphic to $K(T)$. Conversely, $K(T)$ has tree width at most two for every valid $T \in$ tree$(2^{\mathsf{P}}, \mathsf{A})$.*

Recall that our aim is to convert the ICPDL formula φ whose satisfiability is to be decided into an ICPDL formula φ' over P and A and a TWAPTA $\mathcal{T}$. The TWAPTA $\mathcal{T}$ is defined such that it accepts the set of all valid 2^{P}-labeled A-trees. Such a TWAPTA is easily designed, and details are left to the reader. To define the formula φ', we first introduce the auxiliary program

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

and set $\pi_\sim := t? \circ (\pi^1_\sim)^*$. It is easy to see that for each valid tree $T \in \mathrm{tree}(2^{\mathsf{P}}, \mathsf{A})$, we have $[\![\pi_\sim]\!]_T$ equals $\sim$. For an ICPDL program or formula α, let $\widehat{\alpha}$ be obtained from α by replacing

- every atomic program $\gamma \in \mathsf{A}(\varphi) \cap \mathbb{A}$ by

$$\widehat{\gamma} \quad := \quad \bigcup_{i,j \in \{0,1,2\}} \pi_\sim \circ \overline{i} \circ (i,\gamma,j)? \circ j \circ \pi_\sim \qquad \text{and}$$

- every atomic proposition $p \in \mathsf{P}(\varphi) \cap \mathbb{P}$ by $\widehat{p} = \langle \pi_\sim \rangle p$.

Observe that the definitions of $\widehat{\gamma}$ and $\widehat{p}$ directly reflect the definition of $K(T)$. The proof of the following lemma is straightforward by induction on the structure of ψ and π. The base case is easy by definition of $\sim$ and $\pi_\sim$ and $K(T)$, and the inductive step is simple. Details are left to the reader.

Lemma 4. *For all subformulas ψ of φ, subprograms π of φ, and valid trees $T \in tree(2^{\mathsf{P}}, \mathsf{A})$, and places $u, v \in P_T$, we have:*

1. *$u \in [\![\widehat{\psi}]\!]_T$ if and only if $[u] \in [\![\psi]\!]_{K(T)}$;*
2. *$(u, v) \in [\![\widehat{\pi}]\!]_T$ if and only if $([u], [v]) \in [\![\pi]\!]_{K(T)}$.*

Now define $\varphi' := \langle (a \cup b)^* \circ (0 \cup 1 \cup 2) \circ t? \rangle \widehat{\varphi}$.

Lemma 5. *φ is satisfiable if and only if φ' is satisfiable with respect to $\mathcal{T}$.*

Thus, we obtain the following Theorem.

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

5 ω-regular tree satisfiability is in 2EXP

The rest of this section is devoted to showing that ω-regular tree satisfiability in ICPDL is in 2EXP. A matching 2EXP lower bound of ω-regular tree satisfiability in ICPDL follows easily from a result in [12], where it is shown that satisfiability in IPDL over trees is already 2EXP-hard. Therefore, we concentrate on the upper bound for ω-regular tree satisfiability in ICPDL.

We prove containment in 2EXP by an exponential time reduction to the non-emptiness problem for TWAPTAs. The main ingredient of the reduction is a mutual inductive translation of (i) ICPDL formulas into TWAPTAs and (ii) of ICPDL programs into a certain kind of non-deterministic automata (NFAs). The latter resemble standard NFAs on words, but navigate in a complete Σ_E-tree reading symbols from $\{a, \overline{a} \mid a \in \Sigma_E\}$. They can also make conditional ε-transitions, which are executable only if the current tree node is accepted by a given TWAPTA.

Formally, a *non-deterministic finite automaton (NFA) $\mathcal{A}$ over a TWAPTA $\mathcal{T} = (S, \delta, s_0, \mathrm{Acc})$* is a tuple $(Q, p_0, q_0, \rightarrow_{\mathcal{A}})$, where Q is a finite set of *states*, p_0 and q_0 are two *selected states*, and $\rightarrow_{\mathcal{A}}$ is a set of labeled-transitions of the following form, where $q, q' \in Q$ and $a \in \Sigma_E$:

$$q \xrightarrow{a}_{\mathcal{A}} q', \quad q \xrightarrow{\overline{a}}_{\mathcal{A}} q', \quad \text{or} \quad q \xrightarrow{\mathcal{T},s}_{\mathcal{A}} q' \text{ with } s \in S.$$

Transitions of the third kind are called *test transitions*. NFAs define binary relations on the set of nodes of a complete Σ_N-labeled Σ_E-tree. To make this

explicit, let $T \in \text{tree}(\Sigma_N, \Sigma_E)$ and define $\Rightarrow_{\mathcal{A},T} \subseteq (\Sigma_E^* \times Q) \times (\Sigma_E^* \times Q)$ as the smallest relation such that for all $u \in \Sigma_E^*$, $a \in \Sigma_E$, $p, q \in Q$, and $s \in S$, we have

$$(u, p) \Rightarrow_{\mathcal{A},T} (ua, q) \text{ if } p \xrightarrow{a}_{\mathcal{A}} q, \tag{1}$$

$$(ua, p) \Rightarrow_{\mathcal{A},T} (u, q) \text{ if } p \xrightarrow{\overline{a}}_{\mathcal{A}} q, \tag{2}$$

$$(u, p) \Rightarrow_{\mathcal{A},T} (u, q) \text{ if } p \xrightarrow{\mathcal{T},s}_{\mathcal{A}} q \text{ and } (T, u) \in [\![\mathcal{T}, s]\!]. \tag{3}$$

Define

$$[\![\mathcal{A}]\!] = \{(T, u, v) \mid T \in \text{tree}(\Sigma_N, \Sigma_E),\ u, v \in \Sigma_E^*, \text{ and } (u, p_0) \Rightarrow^*_{\mathcal{A},T} (v, q_0)\}.$$

When considering an NFA over a certain TWAPTA $\mathcal{T}$, the initial state of $\mathcal{T}$ is obviously useless. Thus, in any TWAPTA over an NFA will formally be 3-tuple.

5.1 From ICPDL to automata

We show how to convert ICPDL formulas into TWAPTAs and ICPDL programs into NFAs. To this end, fix a finite set of atomic propositions $\mathsf{P} \subseteq \mathbb{P}$ and atomic programs $\mathsf{A} \subseteq \mathbb{A}$ over which ICPDL formulas and programs are built. As expected, the corresponding TWAPTAs and NFAs will work on 2^{P}-labeled A-trees. For ICPDL formulas ψ and programs π, let

$$[\![\psi]\!] = \{(T, u) \mid T \in \text{tree}(2^{\mathsf{P}}, \mathsf{A}),\ u \in \mathsf{A}^*, \text{ and } u \in [\![\psi]\!]_T\}$$
$$[\![\pi]\!] = \{(T, u, v) \mid T \in \text{tree}(2^{\mathsf{P}}, \mathsf{A}),\ u, v \in \mathsf{A}^*, \text{ and } (u, v) \in [\![\pi]\!]_T\}.$$

The aim of this subsection is to convert

- each formula ψ into a TWAPTA $\mathcal{T}(\psi)$ such that $[\![\mathcal{T}(\psi)]\!] = [\![\psi]\!]$ and
- each program π into a TWAPTA $\mathcal{T}(\pi)$ and an NFA $\mathcal{A}(\pi)$ over $\mathcal{T}(\pi)$ such that $[\![\mathcal{A}(\pi)]\!] = [\![\pi]\!]$.

All automata constructed in this section work over 2^{P}-labeled A-trees. The construction is by induction on the structure of ψ and π. We start with defining the TWAPTA $T(\psi)$ for each formula ψ.

If $\psi = p \in \mathsf{P}$, we put $\mathcal{T}(\psi) = (\{s_0\}, \delta, s_0, s_0 \mapsto 1)$, where for all $\gamma \subseteq \mathsf{P}$ we have $\delta(s_0, \gamma) = \texttt{true}$ if $p \in \gamma$ and $\delta(s_0, \gamma) = \texttt{false}$ otherwise. Then clearly we have $[\![\mathcal{T}(\psi)]\!] = [\![\psi]\!]$.

If $\psi = \neg\theta$, then $\mathcal{T}(\psi)$ is obtained from $\mathcal{T}(\theta)$ by applying the standard complementation procedure where all positive Boolean formulas on 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, see e.g. [15].

If $\psi = \langle\pi\rangle\theta$, then we have inductively constructed an NFA

$$\mathcal{A}(\pi) = (Q, p_0, q_0, \rightarrow_{\mathcal{A}}) \text{ over a TWAPTA } \mathcal{T}(\pi) = (S_1, \delta_1, \text{Acc}_1)$$

such that $[\![\pi]\!] = [\![\mathcal{A}(\pi)]\!]$. We have also constructed a TWAPTA

$$\mathcal{T}(\theta) = (S_2, \delta_2, s_2, \mathrm{Acc}_2)$$

such that $[\![\theta]\!] = [\![\mathcal{T}(\theta)]\!]$. We may assume that Q, S_1, and S_2 are pairwise disjoint. We construct the TWAPTA $\mathcal{T}(\psi) = (S, \delta, p_0, \mathrm{Acc})$ with $S = Q \cup S_1 \cup S_2$. For states in S_1 and S_2, the transitions of $\mathcal{T}(\psi)$ are as in $\mathcal{T}(\pi)$ and $\mathcal{T}(\theta)$, respectively. To simulate $\mathcal{A}(\pi)$, we first note that handling transitions $q \xrightarrow{a}_{\mathcal{A}(\pi)} r$ and $q \xrightarrow{\overline{a}}_{\mathcal{A}(\pi)} r$ is easy: $\mathcal{T}(\psi)$ simply navigates up and down the tree as required. To handle a transition $q \xrightarrow{\mathcal{T}(\pi),s}_{\mathcal{A}(\pi)} r$, we branch universally to simulate $\mathcal{T}(\pi)$ in state s *and* to simulate $\mathcal{A}(\pi)$ in state r. To ensure that the simulation of $\mathcal{A}(\pi)$ terminates, the priority function of $\mathcal{T}(\psi)$ assigns 1 to all states of Q. To start the simulation of $\mathcal{T}(\theta)$ after termination, we admit an ε-transition from q_0 to s_2. Formally, for $q \in Q$ and $\gamma \subseteq \mathsf{P}$, we define

$$\delta(q,\gamma) = \bigvee\{\langle r, a\rangle \mid q \xrightarrow{a}_{\mathcal{A}(\pi)} r\} \vee \bigvee\{\langle r, \overline{a}\rangle \mid q \xrightarrow{\overline{a}}_{\mathcal{A}(\pi)} r\} \;\vee$$
$$\bigvee\{\langle s, \varepsilon\rangle \wedge \langle r, \varepsilon\rangle \mid q \xrightarrow{\mathcal{T}(\pi),s}_{\mathcal{A}(\pi)} r\}$$

with an additional disjunct $\langle s_2, \varepsilon\rangle$ if $q = q_0$. The priority function Acc is defined by setting $\mathrm{Acc}(s) = 1$ if $s \in Q$ and $\mathrm{Acc}(s) = \mathrm{Acc}_i(s)$ whenever $s \in S_i$ with $i \in \{1, 2\}$. It is straightforward to check that $[\![\mathcal{T}(\psi)]\!] = [\![\psi]\!]$.

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

If $\pi = a$ or $\pi = \overline{a}$, $a \in \mathsf{A}$, the NFA $\mathcal{A} = \mathcal{A}(\pi)$ has its only transition between its two special states p_0 and q_0, namely $p_0 \xrightarrow{a}_{\mathcal{A}} q_0$ or $p_0 \xrightarrow{\overline{a}}_{\mathcal{A}} q_0$, respectively. Clearly, $[\![\pi]\!] = [\![\mathcal{A}(\pi)]\!]$. Since $\mathcal{A}(\pi)$ has no test transitions, the TWAPTA $\mathcal{T}(\pi)$ is not important. For estimating the size of the constructed automata, we assume that $\mathcal{T}(\pi)$ has a single state and its index is $|\mathrm{Acc}| = 1$.

If $\pi = \psi?$, we can assume that there exists a TWAPTA $\mathcal{T}(\psi) = (S, \delta, s_0, \mathrm{Acc})$ such that $[\![\psi]\!] = [\![\mathcal{T}(\psi)]\!]$. The TWAPTA $\mathcal{T}(\pi)$ is $\mathcal{T}(\psi)$ (without the initial state). The NFA $\mathcal{A}(\pi)$ has only the two special states p_0 and q_0 with the transition $p_0 \xrightarrow{\mathcal{T}(\pi),s_0} q_0$. Hence, we have $[\![\pi]\!] = [\![\mathcal{A}(\pi)]\!] = \{(T, u, u) \mid (T, u) \in [\![\mathcal{T}(\psi)]\!]\}$.

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

It remains to construct $\mathcal{A}(\pi_1 \cap \pi_2)$ and $\mathcal{T}(\pi_1 \cap \pi_2)$, which is the most difficult step of the construction. Assume that the NFA

$$\mathcal{A}(\pi_i) = (Q_i, p_i, q_i, \rightarrow_{\mathcal{A}(\pi_i)}) \text{ over the TWAPTA } \mathcal{T}(\pi_i) \tag{4}$$

has already been constructed, for $i \in \{1, 2\}$. Thus, $[\![\mathcal{A}(\pi_i)]\!] = [\![\pi_i]\!]$. A natural idea for defining an NFA for $\pi_1 \cap \pi_2$ is to apply a product construction to $\mathcal{A}(\pi_1)$

and $\mathcal{A}(\pi_2)$. However, a naive attempt to do this fails: the product construction forces $\mathcal{A}(\pi_1)$ and $\mathcal{A}(\pi_2)$ to travel along the same path, whereas $\mathcal{A}(\pi_1)$ and $\mathcal{A}(\pi_2)$ may actually travel from a tree node u to a tree node v along different paths. More precisely, the two automata both travel along the unique shortest path P from u to v in the tree, but they may make different "detours" from this path P, i.e., they may divert from P and eventually return to the node on P where the diversion started. In order to eliminate this problem, we modify $\mathcal{A}(\pi_1)$ and $\mathcal{A}(\pi_2)$ by admitting additional test transitions that allow to short-cut the described detours. These modified NFA can always travel along the shortest path without any detours, and thus the product construction is applicable.

Before we construct $\mathcal{A}(\pi_1 \cap \pi_2)$, we show how to modify an NFA in order to short-cut detours. Let

$$\mathcal{A} = (Q, q_0, p_0, \rightarrow_{\mathcal{A}}) \tag{5}$$

be an NFA over an TWAPTA

$$\mathcal{T} = (S, \delta, \mathrm{Acc}) \tag{6}$$

and $T \in \mathrm{tree}(2^{\mathsf{P}}, \mathsf{A})$. We define a relation $\mathrm{loop}_{\mathcal{A},T} \subseteq \mathsf{A}^* \times Q \times Q$ describing detours. Intuitively, $(u, p, q) \in \mathrm{loop}_{\mathcal{A},T}$ means that in the tree T, $\mathcal{A}$ can do a detour starting from u in state p and ending at u in state q. Formally, $\mathrm{loop}_{\mathcal{A},T}$ is the smallest set such that:

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

It can be proven that $\mathrm{loop}_{\mathcal{A},T}$ is as required.

Lemma 6. *We have $(u, p, q) \in \mathrm{loop}_{\mathcal{A},T}$ if and only if $(u, p) \Rightarrow^*_{\mathcal{A},T} (u, q)$.*

Since Conditions (i)–(v) can easily be translated into a TWAPTA, we obtain the following.

Lemma 7. *For the NFA $\mathcal{A}$ in (5) over the TWAPTA $\mathcal{T}$ in (6) there is a TWAPTA $\mathcal{U} = (S', \delta', \mathrm{Acc}')$ with $S' = S \uplus (Q \times Q)$ such that for all $s \in S$ and $p, q \in Q$:*

(i) $[\![\mathcal{U}, s]\!] = [\![\mathcal{T}, s]\!]$,
(ii) $[\![\mathcal{U}, (p, q)]\!] = \{(T, u) \mid T \in \mathit{tree}(2^{\mathsf{P}}, \mathsf{A}), u \in \mathsf{A}^*, \text{ and } (u, p) \Rightarrow^*_{\mathcal{A},T} (u, q)\}$, *and*
(iii) $|\mathrm{Acc}'| = |\mathrm{Acc}|$.

Note that the TWAPTA $\mathcal{U}$ can be seen as an extension of the basic TWAPTA $\mathcal{T}$ in (6).

Recall that our aim is to modify the NFA $\mathcal{A}$ from in (5) such that detours are short-cut. To do this, define the new NFA $\mathcal{B} = (Q, p_0, q_0, \rightarrow_{\mathcal{B}})$ over the TWAPTA

$\mathcal{U}$ as the modification of $\mathcal{A}$ obtained by adding for every pair $(p,q) \in Q \times Q$, the test transition $p \xrightarrow{\mathcal{U},(p,q)}_{\mathcal{B}} q$. For words $u, v \in \mathsf{A}^*$ 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. Define the relation $\Uparrow_{\mathcal{B},T} \subseteq (\mathsf{A}^* \times Q) \times (\mathsf{A}^* \times Q)$ (resp. $\Downarrow_{\mathcal{B},T} \subseteq (\mathsf{A}^* \times Q) \times (\mathsf{A}^* \times Q)$) in the same way as $\Rightarrow_{\mathcal{B},T}$, except that clause (1) (resp. (2)), with $\mathcal{A}$ replaced by $\mathcal{B}$, in the definition of the relation $\Rightarrow_{\mathcal{B},T}$ is dropped. This means that with the relation $\Uparrow_{\mathcal{B},T}$ (resp. $\Downarrow_{\mathcal{B},T}$) the NFA $\mathcal{B}$ is only allowed to walk up (resp. down) in the tree or stay in the same node by executing a test transition.

Lemma 8. *Let $u, v \in \mathsf{A}^*$. Then the following three statements are equivalent:*

(i) $(u,v) \in [\![\mathcal{A}]\!]$
(ii) $(u,v) \in [\![\mathcal{B}]\!]$
(iii) there exists $r \in Q$ with $(u, p_0) \Uparrow^*_{\mathcal{B},T} (\inf(u,v), r) \Downarrow^*_{\mathcal{B},T} (v, q_0)$.

We now return to the construction of $\mathcal{A}(\pi_1 \cap \pi_2)$ and $\mathcal{T}(\pi_1 \cap \pi_2)$ from the NFA $\mathcal{A}_i = \mathcal{A}(\pi_i)$ over the TWAPTA $\mathcal{T}_i = \mathcal{T}(\pi_i)$ in (4) ($i \in \{1,2\}$). For $i \in \{1,2\}$, we convert $\mathcal{T}_i$ into a new TWAPTA $\mathcal{U}_i$ as in Lemma 7 and $\mathcal{A}_i$ into a new NFA $\mathcal{B}_i = (Q_i, p_i, q_i, \rightarrow_{\mathcal{B}_i})$ over $\mathcal{U}_i$ as described above. Define $\mathcal{T}(\pi_1 \cap \pi_2)$ as the disjoint union of $\mathcal{U}_1$ and $\mathcal{U}_2$. The NFA $\mathcal{A}(\pi_1 \cap \pi_2)$ is the product automaton of $\mathcal{B}_1$ and $\mathcal{B}_2$, where test transitions can be carried out asynchronously:

$$\mathcal{A}(\pi_1 \cap \pi_2) \quad := \quad (Q_1 \times Q_2, (p_1,p_2), (q_1,q_2), \rightarrow_{\mathcal{A}(\pi_1 \cap \pi_2)}),$$

where $\rightarrow_{\mathcal{A}(\pi_1 \cap \pi_2)}$ is the smallest relation such that

- $r_1 \xrightarrow{a}_{\mathcal{B}_1} r_1'$ and $r_2 \xrightarrow{a}_{\mathcal{B}_2} r_2'$ implies $(r_1, r_2) \xrightarrow{a}_{\mathcal{A}(\pi_1 \cap \pi_2)} (r_1', r_2')$, and analogously for $\overline{a}$-transitions,
- $r_1 \xrightarrow{\mathcal{U}(\pi_1),s}_{\mathcal{B}_1} r_1'$ implies $(r_1, r_2) \xrightarrow{\mathcal{T}(\pi_1 \cap \pi_2),s}_{\mathcal{A}(\pi_1 \cap \pi_2)} (r_1', r_2)$ for all $r_2 \in Q_2$,
- $r_2 \xrightarrow{\mathcal{U}(\pi_2),s}_{\mathcal{B}_2} r_2'$ implies $(r_1, r_2) \xrightarrow{\mathcal{T}(\pi_1 \cap \pi_2),s}_{\mathcal{A}(\pi_1 \cap \pi_2)} (r_1, r_2')$ for all $r_1 \in Q_1$.

Lemma 9. $[\![\mathcal{A}(\pi_1 \cap \pi_2)]\!] = [\![\pi_1 \cap \pi_2]\!]$.

This finishes the inductive translation of ICPDL formulas and programs into automata.

5.2 Automata size

We analyze the size of the automata constructed in the previous subsection. It will turn out that our translation of formulas and programs into automata has an exponential blow-up. Next, we introduce a syntactic parameter which the size of the constructed TWAPTAs is exponential in, namely intersection width. Formally, the *intersection width* $\mathrm{iw}(\pi)$ of an ICPDL program π is defined

inductively:

$$\begin{aligned} \mathrm{iw}(a) = \mathrm{iw}(\overline{a}) &= 1 \text{ for all } \ a \in \mathsf{A} \\ \mathrm{iw}(\theta?) &= 1 \\ \mathrm{iw}(\pi_1 \cup \pi_2) = \mathrm{iw}(\pi_1 \circ \pi_2) &= \max\{\mathrm{iw}(\pi_i), \mathrm{iw}(\pi_2)\} \\ \mathrm{iw}(\pi^*) &= \mathrm{iw}(\pi) \\ \mathrm{iw}(\pi_1 \cap \pi_2) &= \mathrm{iw}(\pi_1) + \mathrm{iw}(\pi_2) \end{aligned}$$

Note that iw is non-montonic: $\mathrm{iw}(\theta?) = 1$ but $\theta?$ may contain subprograms of intersection width stictly larger than 1.

For an NFA $\mathcal{A} = (Q, p_0, q_0, \rightarrow_{\mathcal{A}})$ (over some TWPATA) we define the size of $\mathcal{A}$ as $|Q|$. Note that for the size of $\mathcal{A}$, the size of the TWPATA over which it is defined, is not taken into account.

Lemma 10. *For every ICPDL program π, $|\mathcal{A}(\pi)| \leq 2|\pi|^{\mathrm{iw}(\pi)}$.*

Recall that intersection width is non-monotonic. In order to estimate the size of TWAPTA, we need a monotonic variant: If α is either an ICPDL program or an ICPDL formula then

$$\mathrm{IW}(\alpha) \quad = \quad \begin{cases} 1 & \text{if no subprogram occurs in } \alpha \\ \max\{\mathrm{iw}(\pi) \mid \pi \text{ occurs in } \alpha\} & \text{else.} \end{cases}$$

Recall that for a TWAPTA $\mathcal{T}$ we denote by $i(\mathcal{T})$ the index of $\mathcal{T}$.

Lemma 11. *For every ICPDL program π and ICPDL formula ψ:*

- $|\mathcal{T}(\pi)| \leq |\pi| + 8 \cdot |\pi| \cdot |\pi|^{2 \cdot \mathrm{IW}(\pi)}$ *and* $i(\mathcal{T}(\pi)) \leq |\pi|$.
- $|\mathcal{T}(\psi)| \leq |\psi| + 8 \cdot |\psi| \cdot |\psi|^{2 \cdot \mathrm{IW}(\psi)}$ *and* $i(\mathcal{T}(\psi)) \leq |\psi|$.

By applying Lemma 11 and Theorem 2, we obtain the following upper bound on ω-regular tree satisfiability in ICPDL.

Theorem 5. *For a TWAPTA $\mathcal{T}$ and an ICPDL formula φ, we can decide in time $2^{(|\mathcal{T}|+|i(\mathcal{T})|+|\varphi|)^{\mathrm{IW}(\varphi)\,O(1)}}$ whether there exists some $T \in L(\mathcal{T})$ such that $\varepsilon \in [\![\varphi]\!]_T$. Hence, ω-regular tree satisfiability belongs to* 2EXP.

Note that Theorem 5 yields a single exponential upper complexity bound if the intersection width of input formulas φ is bounded by a constant.

Thus, by Theorem 4 and Theorem 5, we obtain our main result.

Theorem 6. *Satisfiability in ICPDL is* 2EXP*-complete. For every constant $c \in \mathbb{N}$, satisfiability in $\{\varphi \mid \varphi$ is an ICPDL formula with $\mathrm{IW}(\varphi) \leq c\}$ is* EXP*-complete.*

Consider the following extension of PDL with converse that we call loop-CPDL: the loop-operator gives for a program π a formula $\mathrm{loop}(\pi)$ with the following semantics: $[\![\mathrm{loop}(\pi)]\!]_K = \{x \mid (x, x) \in [\![\pi]\!]_K\}$. Note that loop can be defined using $\cap$: $[\![\mathrm{loop}(\pi)]\!]_K = [\![\langle \pi \cap \mathtt{true}? \rangle \mathtt{true}]\!]_K$.

Corollary 2. *Satisfiability of* loop*-CPDL is* EXP*-complete.*

6 Open problems

An interesting open problem is the *finite satisfiability problem for I(C)PDL*. Here, for a given I(C)PDL formula φ it is asked whether φ has a *finite* model. It is currently even open, whether the finite satisfiability problem for IPDL is decidable. Bojańczyk proved in [3] that finite satisfiability for the modal μ-calculus with backwards modalities (another logic, which does not have the finite model property) is EXP-complete. Maybe one can use techniques from that paper in order to attack the finite satisfiability problem for I(C)PDL.

References

1. L. Afanasiev, P. Blackburn, I. Dimitriou, E. G. Bertrabd Gaiffe, M. Marx, and M. de Rijke. PDL for ordered trees. *Journal of Applied Non-Classical Logics*, 15(2):115–135, 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. M. Bojańczyk. Two-way alternating automata and finite models. In *29th International Colloquium on Automata, Languages and Programming (ICALP 2002), Malaga (Spain)*, number 2380 in Lecture Notes in Computer Science, pages 833–844. Springer, 2002.
4. R. Danecki. Nondeterministic propositional dynamic logic with intersection is decidable. In *Proceedings of the 5th Symposium on Computation Theory (Zaborw, Poland)*, number 208 in Lecture Notes in Computer Science, pages 34–53, 1984.
5. R. Danecki. Propositional Dynamic Logic with Strong Loop Predicate. In *Proceedings of Mathematical Foundations of Computer Science 1984 (MFCS 1984)*, number 176 in Lecture Notes in Computer Science, pages 573–581, 1984.
6. L. Farinas Del Cerro and E. Orlowska. DAL-a logic for data analysis. *Theoretical Computer Science*, 36(2-3):251–264, 1985.
7. M. J. Fischer and R. E. Ladner. Propositional Dynamic Logic of Regular Programs. *Journal of Computer and System Sciences*, 18(2):194–211, 1979.
8. J. Flum. On the (infinite) model theory of fixed point logics. In X. Caicedo and C. H. Montenegro, editors, *Models, Algebras, and Proofs: selected papers of the X Latin American symposium on mathematical logic held in Bogota*, volume 203 of *Lecture Notes in Pure and Applied Mathematics*, pages 67–75. Marcel Dekker, Inc., 1999.
9. G. D. Giacomo and M. Lenzerini. Boosting the Correspondence between Description Logics and Propositional Dynamic logics. In *Proceedings of the 12th National Conference on Artifical Intelligence (AAAI'94)*, pages 205–212, 1994.
10. S. Göller, M. Lohrey, and C. Lutz. PDL with Intersection and Converse Is 2 EXP-Complete. In *Proceedings of the 10th International Conference on Foundations of Software Science and Computational Structures (FoSSaCS 2007), Braga (Portugal)*, number 4423 in Lecture Notes in Computer Science, pages 198–212. Springer, 2007.
11. D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. Foundations of computing. The MIT Press, 2000.
12. M. Lange and C. Lutz. 2-ExpTime Lower Bounds for Propositional Dynamic Logics with Intersection. *Journal of Symbolic Logic*, 70(4):1072–1086, 2005.

13. C. Lutz. PDL with Intersection and Converse Is Decidable. In C.-H. L. Ong, editor, *Proceedings of the 19th International Workshop on Computer Science Logic (CSL 2005), Oxford (UK)*, number 3634 in Lecture Notes in Computer Science, pages 413–427. Springer, 2005.
14. 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.
15. D. Muller and P. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54(2-3):267–276, 1987.
16. V. Pratt. A near-optimal method for reasoning about action. *Journal of Computer and System Sciences*, 20:231–254, 1980.
17. 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.
18. B. ten Cate and C. Lutz. The Complexity of Query Containment in Expressive Fragments of XPath 2.0. In *Proceedings of the 26th ACM Symposium on Principles of Database Systems (PODS'98)*. ACM Press, 2007.
19. W. van der Hoek and J. Meyer. A complete epistemic logic for multiple agents - combining distributed and common knowledge, 1997.
20. H. P. van Ditmarsch, W. van der Hoek, and B. P. Kooi. Concurrent dynamic epistemic logic for MAS. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2003)*, pages 201–208. ACM Press, 2003.
21. M. Y. Vardi. The taming of converse: Reasoning about two-way computations. In *Proceedings of Logics of Programs*, number 193 in Lecture Notes in Computer Science, pages 413–423. Springer, 1985.
22. M. Y. Vardi. Reasoning about The Past with Two-Way Automata. In K. G. Larsen, S. Skyum, and G. Winskel, editors, *Proceedings of the 25th International Colloquium on Automata, Languages and Programming (ICALP '98), Aalborg (Denmark)*, number 1443 in Lecture Notes in Computer Science, pages 628–641. Springer, 1998.