Baby Steps towards Finite Satisfiability for LoopPDL

Bartosz Jan Bednarczyk[1,2], Mikołaj Swoboda[2]

[1]Institute of Logic and Computation, TU Wien
[2]Computer Science Department, University of Wrocław

Abstract

Propositional Dynamic Logic (PDL), known in the Description Logic community as $\mathcal{ALC}_{\text{reg}}$, is a well-established modal logic of programs. To increase its expressive power, several extensions have been proposed. One such extension is the loop operator, which expresses that an element can reach itself via a path defined by a regular expression. Although the satisfiability problem for LoopPDL is well understood, the decidability of its finite satisfiability problem has remained open for over 40 years.

Our ongoing work aims to resolve this question. While a full solution is still pending, we have obtained some results of independent interest. In this workshop paper, we present a simplified version of our approach, applied to a logic with loops restricted to atomic roles and their transitive closures.

Keywords

finite-model reasoning, satisfiability, complexity, regular path expressions

1. Introduction

PDL. Propositional Dynamic Logic (PDL) is a well-established modal and description logic for reasoning about program behavior. Introduced by Fischer and Ladner [1] in the late 1970s, it has since inspired extensive research on the complexity, expressiveness, proof theory, and model theory of dynamic logics; see the surveys by Harel et al. [2] and Troquard and Balbiani [3] for further details. From the perspective of graph databases, PDL can also be viewed as a unifying framework combining ontology-mediated data access with expressive graph query languages such as regular path queries. In essence, PDL extends the classical description logic $\mathcal{ALC}$ with modalities indexed by *programs* s (i.e., regular path expressions). These are regular expressions over an alphabet of atomic programs (roles) $r \in \mathbf{N_R}$ and tests of the form A? for atomic concepts $\mathrm{A} \in \mathbf{N_C}$ (and, more generally, C? for complex concepts C). Semantically, an element a in an interpretation $\mathcal{I}$ satisfies $\exists s.\mathrm{C}$ if it can reach an element satisfying C via a path whose labels match s, where tests D? are interpreted as self-loops on elements satisfying D. Fischer and Ladner showed that PDL enjoys the finite model property (FMP) [1, Thm. 3.2]—every satisfiable formula has a finite model. Their work, together with that of Pratt, determined that its satisfiability problem is ExpTime-complete [1, Thm. 4.4][4]. Analogous results hold for CPDL (converse PDL), as well as for many expressive description-logic generalizations such as $\mathcal{ZOI}$ [5, Thm. 6] and $\mathcal{ZOQ}$ [6, Thm. 3.1].

Extensions of PDL. Despite its expressive power, PDL cannot capture cyclic program behaviour. To overcome this limitation, Danecki [7] extended PDL in 1984 with the $\exists s.\mathsf{Self}$ operator, yielding the logic LoopPDL. The semantics of this operator is as expected: an element satisfies $\exists s.\mathsf{Self}$ if it can reach itself via a path matching s. Note that, in contrast to the standard Self operator used in the description logic community (e.g. in $\mathcal{SROIQ}$), the programs occurring in loops need not be atomic. Danecki also established ExpTime-completeness of the satisfiability problem for LoopPDL [7, Thm. 1] using an automata-theoretic argument based on the "cactus model property" of the logic. This line of work was further generalized by Harel [8, p. 182], who enriched PDL with program intersection $\cap$ (parallel execution of programs), a construct that was studied intensively in the early 2000s [9, 10, 11, 12, 13].

DL 2026: 39th International Workshop on Description Logics, July 17–19, 2026, Lisbon, Portugal
bartek@cs.uni.wroc.pl (B. J. Bednarczyk); 331319@uwr.edu.pl (M. Swoboda)
https://bartoszjanbednarczyk.github.io (B. J. Bednarczyk)
0000-0002-8267-7554 (B. J. Bednarczyk)

Finite-model reasoning. In stark contrast to PDL and its many description-logic generalizations [5, 6], even very small fragments of LoopPDL fail to enjoy the FMP. Indeed, consider the TBox

$$\top \sqsubseteq \exists r.\top \sqcap \neg \exists r^{+}.\mathsf{Self},$$

which states that every element has a successor, but no element lies on a cycle. This TBox has a model (e.g. an infinite chain) but no finite models. This observation motivates the study of the finite satisfiability problem for LoopPDL and its fragments. Decidability of finite satisfiability for LoopPDL (and its generalizations with $\cap$ operator) has been posed as an open problem in numerous works [14, p. 12][13, §7][15, Sec. 3.8][16, Open question 1][17, p. 94]. Somewhat surprisingly, to the best of our knowledge, no positive results are known even for the single-role fragment of $\mathcal{ALC}$ extended with both reachability and self-reachability, namely the restriction of LoopPDL to a single role r and its transitive closure r^{+}. Informally, the main obstacle in addressing these questions is the presence of cycles in the intended models—expressible in these logics—together with the non-trivial interactions between different cycles under the finiteness constraint. In the words of Gottlob's well-known motto:

"*All the evil in computer science comes from cycles.*"

Positive results. We briefly survey several positive results for rather restricted (and somewhat artificial) fragments of LoopPDL that can be extracted from the literature. To the best of our knowledge, no other fragments of our logic are currently known to admit a decidable finite satisfiability problem, rendering the state of the art rather unsatisfactory. Moreover, the available complexity bounds for the last two fragments discussed below are not tight.

- The *atomic-loop fragment* of LoopPDL, i.e., PDL extended with the Self operator restricted to atomic programs. This logic is a fragment of $\mathcal{ZOI}$ and, as noted above, enjoys the finite model property.
- The *transitive fragment* of LoopPDL, i.e., a sublogic that allows programs that are either atomic or transitive roles. This logic can be encoded into the two-variable guarded fragment with transitive guards, which is known to have a decidable finite satisfiability problem [18, Thm. 5.9].
- The *query fragment* of LoopPDL, i.e., a sublogic in which (i) regular expressions may only occur within the Self operator, and (ii) the Self operator appears only under an odd number of negations. This logic can be reduced to the finite entailment problem for conjunctive regular path queries under $\mathcal{ALC}$-TBoxes, which was recently shown to be decidable [19, Thm. 1]. In the PhD thesis of Gutowski [20, Thm. 1.1] it was shown that the restriction (i) is not needed for decidability. Note that this sublogic allows for "forbidding cycles" but it is not capable of expressing/enforcing their existence.

Our motivation and results. Our primary motivation is to address the finite satisfiability problem for ICPDL—PDL extended with converse and intersection, and thus a highly expressive generalization of LoopPDL—a long-standing and notoriously challenging open problem in computational logic that has seen only limited progress over the past four decades. While we do not resolve finite satisfiability in full generality, even for LoopPDL, we obtain results for meaningful fragments that are of independent interest. In this workshop paper, we present a streamlined version of our approach, focusing on a fragment of LoopPDL where loops are restricted to atomic roles and their transitive closures, while programs outside loops remain unrestricted. To the best of our knowledge, this constitutes the *first substantial progress* toward establishing the decidability of the finite satisfiability problem for LoopPDL.

2. Preliminaries

Basics on DLs. We consider disjoint, countably infinite sets of *role names* $\mathbf{N_R}$ and *concept names* $\mathbf{N_C}$. The syntax of our fragment of LoopPDL is defined by the following context-free grammar:

$$\mathrm{C}, \mathrm{C}' ::= \top \mid \mathrm{A} \mid \neg \mathrm{C}' \mid \mathrm{C} \sqcup \mathrm{C}' \mid \exists s.\mathrm{C} \mid \exists r.\mathsf{Self} \mid \exists r^{+}.\mathsf{Self} \qquad \text{(concepts)}$$

$$s, s' ::= r \mid \mathrm{C}? \mid s + s' \mid s \circ s' \mid s^* \qquad \text{(roles)}$$

for $\mathrm{A} \in \mathbf{N_C}$ and $r \in \mathbf{N_R}$. We employ the usual abbreviations for conjunction $\mathrm{C} \sqcap \mathrm{D} \coloneqq \neg(\neg\mathrm{C} \sqcup \neg\mathrm{D})$, value restriction $\forall s.\mathrm{C} \coloneqq \neg\exists s.\neg\mathrm{C}$, falsum $\bot \coloneqq \neg\top$, and transitive closure $s^+ \coloneqq s \circ s^*$ of roles. Note that in contrast to the complete syntax of LoopPDL, we do not allow the Self operator to be applied to arbitrary roles, but only to (possibly transitively closed) role names. Nevertheless, all roles may occur in existential and value restrictions.

Table 1
Concepts in LoopPDL.

Name	Syntax	Semantics
top concept	$\top$	$\Delta^{\mathcal{I}}$
concept name	A	$\mathrm{A}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
conc. negation	$\neg\mathrm{C}$	$\Delta^{\mathcal{I}} \setminus \mathrm{C}^{\mathcal{I}}$
conc. union	$\mathrm{C} \sqcup \mathrm{D}$	$\mathrm{C}^{\mathcal{I}} \cup \mathrm{D}^{\mathcal{I}}$
exist. restriction	$\exists s.\mathrm{C}$	$\{d \mid \exists e.(d,e) \in s^{\mathcal{I}} \wedge e \in \mathrm{C}^{\mathcal{I}}\}$
Self operator	$\exists s.\mathsf{Self}$	$\{d \mid (d,d) \in s^{\mathcal{I}}\}$

Table 2
Roles in LoopPDL.

Name	Syntax	Semantics
role name	r	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
test	$\mathrm{C}?$	$\{(d,d) \mid d \in \mathrm{C}^{\mathcal{I}}\}$
role comp.	$s \circ s'$	$s^{\mathcal{I}} \circ (s')^{\mathcal{I}}$
role union	$s + s'$	$s^{\mathcal{I}} \cup (s')^{\mathcal{I}}$
Kleene's star	s^*	refl. & trans. closure of $s^{\mathcal{I}}$

The semantics of our fragment of LoopPDL is given by interpretations. An *interpretation* is a pair $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ consisting of a non-empty set $\Delta^{\mathcal{I}}$ called the *domain* and a function $\cdot^{\mathcal{I}}$ that maps concept names A to subsets of the domain $\mathrm{A}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and maps role names r to sets of pairs of domain elements $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. This mapping is then extended to arbitrary concepts (Table 1) and roles (Table 2). An interpretation is *finite* if its domain is finite. As usual, a *TBox* $\mathcal{T}$ is a finite set of *concept inclusions*, written $\mathrm{C} \sqsubseteq \mathrm{D}$. We write $(\mathrm{C} \equiv \mathrm{D}) \in \mathcal{T}$ when $(\mathrm{C} \sqsubseteq \mathrm{D}) \in \mathcal{T}$ and $(\mathrm{D} \sqsubseteq \mathrm{C}) \in \mathcal{T}$, and call $\mathrm{C} \equiv \mathrm{D}$ a *concept equivalence*. We use $\mathsf{con}(\mathcal{T})$ and $\mathsf{rol}(\mathcal{T})$ to denote the set of all concept names occurring in $\mathcal{T}$ and the set of all role names occurring in $\mathcal{T}$, respectively. An interpretation $\mathcal{I}$ *satisfies* a concept inclusion $\mathrm{C} \sqsubseteq \mathrm{D}$ (written: $\mathcal{I} \models \mathrm{C} \sqsubseteq \mathrm{D}$) if $\mathrm{C}^{\mathcal{I}} \subseteq \mathrm{D}^{\mathcal{I}}$. Moreover, $\mathcal{I}$ satisfies a TBox $\mathcal{T}$ (written: $\mathcal{I} \models \mathcal{T}$) if $\mathcal{I}$ satisfies all concept inclusions that belong to $\mathcal{T}$. In such case, we call $\mathcal{I}$ a *model* of $\mathcal{T}$. Additionally, a concept C is *(finitely) satisfiable with respect to* $\mathcal{T}$ if there exists a (finite) model $\mathcal{I}$ of $\mathcal{T}$ and an element $a \in \Delta^{\mathcal{I}}$ such that $a \in \mathrm{C}^{\mathcal{I}}$. We may then call $\mathcal{I}$ a *model* of $\langle \mathcal{T}, \mathrm{C} \rangle$. In the (finite) satisfiability problem we ask, given a TBox $\mathcal{T}$ and a concept name C as an input, if C is (finitely) satisfiable with respect to $\mathcal{T}$.

Walks and cycles. A *walk* in an interpretation $\mathcal{I}$ is a finite word $\pi \coloneqq a_0, r_1, a_1, r_2 \ldots, r_{n-1}, a_{n-1}$ in $\Delta^{\mathcal{I}}(\mathbf{N_R} \cdot \Delta^{\mathcal{I}})^*$ such that for all $i \leqslant n-2$ we have $\langle a_i, a_{i+1} \rangle \in r_{i+1}^{\mathcal{I}}$. We say that π starts from a if $a_0 = a$. Analogously, π ends on a if $a_{n-1} = a$. If there is $r \in \mathbf{N_R}$ such that all r_i are equal to r, we call such a π an *r-walk*. A *cycle* (respectively, an r-cycle) is a walk (an r-walk) with $a_{n-1} = a_0$.

Automata. By a *nondeterministic finite automaton* (NFA) $\mathcal{A}$ we understand, as usual, a tuple $(Q, \Sigma, \delta, q_0, F)$ where Q is a finite set of states, Σ is a finite alphabet, $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation, $q_0 \in Q$ is the initial state and $F \subseteq Q$ is the set of accepting states. All NFAs considered in this paper are over finite subsets of $\{r, \mathrm{C}? \mid r \in \mathbf{N_R}, \mathrm{C} \in \mathbf{N_C}\}$, where the expressions of the form C? are called *tests*. If q and q' are states of $\mathcal{A}$, we write $\mathcal{A}_{q,q'}$ to denote the NFA obtained from $\mathcal{A}$ by changing the initial state to q and the set of accepting states to $\{q'\}$.

By a *run* of an NFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ over an interpretation $\mathcal{I}$, we understand a finite sequence of tuples $(d_0, q_0, w_1, q_1, d_1), \ldots, (d_{\ell-1}, q_{\ell-1}, w_\ell, q_\ell, d_\ell)$ in $\Delta^{\mathcal{I}} \times \delta \times \Delta^{\mathcal{I}}$ such that for all $i < \ell$ we have $(d_i, d_{i+1}) \in (w_i)^{\mathcal{I}}$. A run is *accepting* if $q_\ell \in F$. We say that a run starts from d if $d_0 = d$ (and similarly ends on d if $d_\ell = d$). Suppose now that there is an accepting run of $\mathcal{A}$ that starts from d, and let $(d_0, \delta_0, d_1), \ldots, (d_{\ell-1}, \delta_0, d_\ell)$ with $\delta_i = (q_i, w_{i+1}, q_{i+1})$ be the sequence of tuples obtained from the run by removing the ones concerning tests. If the resulting sequence is empty, then by an *$\mathcal{A}$-witness* for d we understand the singleton sequence d. Otherwise, it is the walk $d_0, w_1, d_1, \ldots, w_\ell, d_\ell$. We speak about an $\mathcal{A}$-self-witness for d when there is an accepting run of $\mathcal{A}$ that starts and ends on d. Such walks are called *witnessing walks* and *cycles*, respectively. A similar notation is used for regular expressions.

From now on it will be convenient to work with automata in existential restrictions in place of regular expressions with tests. Hence, we extend the syntax of our logic with the concept constructor $\exists\mathcal{A}.\mathrm{C}$, interpreted by $\mathcal{I}$ as the set of all pairs (a, b) for which there is an accepting run of $\mathcal{A}$ from a to b.

Trees. We use the set-theoretic representation of a (countable, bounded-branching) *tree* as a prefix-closed subset of $\{0, 1, \ldots, N\}^*$ for some $N \in \mathbb{N}$. Elements of a tree are called *vertices*, and the empty sequence ε is the *root*. For a vertex v, its *children* are the vertices of the form vi with $0 \leqslant i \leqslant N$. Analogously, a *descendant* of v is a vertex of the form vw for some $w \in \{0, 1, \ldots, N\}^+$. The notion of a *parent* is defined dually. A (possibly infinite) sequence of vertices $v_0, v_1, \ldots$ such that v_{i+1} is a child of v_i is called a *path*. If there exists j such that v_{i+1} is always the j-child of v_i, then the path is a *j-path*. A *labelled tree* is a tree equipped with a labelling function L mapping vertices to elements of a fixed label set. A tree T' (with labelling L') is a *subtree* of a tree T (with labelling L) if there exists $v \in T$ such that $\{w \mid vw \in T\} = T'$ and $L'(w) = L(vw)$ for all $w \in T'$. A tree is called *regular* if it has only finitely many distinct subtrees.

Normal form. Let $\mathcal{T}$ be a TBox in our fragment of $\mathrm{LoopPDL}$. We say that $\mathcal{T}$ is in *normal form* if there exists an NFA $\mathcal{A}$ for which $\mathcal{T}$ consists only of concept equivalences of the form $\mathrm{D}_1 \equiv \mathrm{D}_2$ where D_1 is a concept name and

(1) D_2 is a Boolean combination of concept names, or

(2) D_2 is of the form $\exists\mathcal{A}_{q,q'}.\top$ for states q, q' of $\mathcal{A}$, or *(walk demand)*

(3) D_2 is of one of the forms $\exists r.\mathsf{Self}$ and $\exists r^+.\mathsf{Self}$ for a role name r. *(cycle demand)*

Equivalences with D_2 as described in case (1) are called *Boolean constraints*. Note that all tests in $\mathcal{A}$ are atomic. What is more, w.l.o.g. we can assume the right-hand sides of the concept equivalences are unique, and that TBoxes in normal form contain concept equivalences $\mathrm{A} \equiv \exists\mathcal{A}_{q,q'}.\top$ for any pair of states q, q' of $\mathcal{A}$. The use of a normal form allows us to simplify the reasoning about the satisfiability of a TBox, by focusing only on the local demands of each of the elements that a model of the TBox must satisfy. By transforming regular expressions in existential restrictions into NFAs and by routine renaming á la Scott we can establish the following lemma (check appendix for details):

Lemma 2.1. *For every instance $\langle\mathcal{T}, \mathrm{A}\rangle$ of the (finite) satisfiability problem, one can compute in polynomial time a TBox $\mathcal{U}$ in normal form, such that A is (finitely) satisfiable with respect to $\mathcal{U}$ if and only if A is (finitely) satisfiable with respect to $\mathcal{T}$.* ◀

For the remainder of the paper, we **fix** an instance $\langle\mathcal{T}, \mathrm{A}\rangle$ of the finite satisfiability problem, with the TBox $\mathcal{T}$ in normal form. For convenience, we also **fix** a number N and an enumeration $\mathrm{A}_1 \equiv \mathrm{C}_1, \ldots, \mathrm{A}_N \equiv \mathrm{C}_N$ of the concept equivalences in $\mathcal{T}$ whose right-hand sides are cycle demands or walk demands. Finally, we **fix** $\mathbf{N}_{\mathbf{R}}^{\circlearrowleft}$ to be the set of role names r such that $\exists r^+.\mathsf{Self}$ appears in $\mathcal{T}$.

3. Tree Representations of Models and the Algorithm

Overview. One of the classical approaches to the satisfiability problem in many modal and description logics is automata-based. The main idea is to represent models of a given instance as infinite trees and to check for the existence of such representations using automata, typically two-way alternating parity tree automata. These tree representations are usually obtained via variants of unravellings, which transform (potentially infinite) models into infinite trees. In our setting, two difficulties arise. First, we are concerned with finite satisfiability, whereas unravelling does not preserve finiteness. Second, cycle demands in TBoxes, which may require cycles of arbitrary length, are not naturally compatible with tree-shaped representations. To address these issues, we introduce a novel form of

tree representation. Although these trees are infinite, they are equipped with constraints ensuring that they can be "wrapped" into finite models. Furthermore, cycle demands are handled by allowing special paths, called *cycle-guaranteeing paths*, which witness such demands and can be folded into cycles by identifying their endpoints. Further details are developed in the remainder of this section.

Encoding. From now on, we assume that all trees in this paper are subsets of $\{0, 1, \dots, N\}^*$ (with N fixed at the end of the previous section) and are equipped with a labelling function L taking values in

$$\Sigma \coloneqq \mathcal{P}(\mathsf{con}(\mathcal{T}) \cup \{\mathrm{in}_r, \mathrm{self}_r \mid r \in \mathsf{rol}(\mathcal{T})\} \cup \{\mathrm{fin}\})$$

with $\mathcal{P}$ denoting a power set. The purpose of these labels will be specified shortly. Intuitively, concept labels encode the interpretation of concept names at vertices, in_r labels record r-connections from parents to children, self_r labels indicate r-self-loops, and fin labels mark the endpoints of dedicated 0-paths, which will later be folded into cycles. In our encoding, for a vertex v labelled with A_i, the i-child (where $1 \leqslant i \leqslant N$) initiates a path that realizes the i-th demand $\mathrm{A}_i \equiv \mathrm{C}_i$ for v. Such paths corresponding to cycle demands are called *cycle-guaranteeing paths*. Then, v is said to *demand* the cycle realized by the path. Otherwise, if v is not labelled with A_i, we say that v *forbids* all cycles that would satisfy C_i. The 0-child of v is used to continue the current guaranteeing path, if one exists. Consequently, every vertex except the root lies on a path incorporating demands of some other node. More specifically, such paths starting at v have the form $v, vi, vi0, \dots, vi0^k$, where i identifies a demand as enumerated at the end of the previous section. Important requirements of our encoding are summarized below.

Definition 3.1. *A labelled tree $\langle T, L\rangle$ is* encoding-ready *if the following conditions hold:*

1. *The root is not labelled with in_r for any r, and every other vertex is labelled with in_r for exactly one r.*
2. *A vertex labelled with fin has no children and carries no other labels except in_r for some r.*
3. *All 0-paths are finite.*
4. *For every i such that C_i is a cycle demand, every path of the form $v, vi, vi0, \dots, vi0^k$, where $vi0^k$ has no 0-children, satisfies $\mathrm{fin} \in L(vi0^k)$ (including the case $k = 0$).* ◀

Such a $\langle T, L\rangle$ can be seen as a cactus-shaped interpretation $\mathcal{I}(T, L) \coloneqq \langle \Delta^{\mathcal{I}(T,L)}, \cdot^{\mathcal{I}(T,L)}\rangle$, where:

- $\Delta^{\mathcal{I}(T,L)} = \{v \in T \mid \mathrm{fin} \notin L(v)\}$,
- for $v \in \Delta^{\mathcal{I}(T,L)}$ and $\mathrm{B} \in \mathsf{con}(\mathcal{T})$, we have $v \in \mathrm{B}^{\mathcal{I}(T,L)}$ if and only if $\mathrm{B} \in L(v)$,
- for each $r \in \mathsf{rol}(\mathcal{T})$, $r^{\mathcal{I}(T,L)}$ is the $\subseteq$-minimal relation such that:
  - for $v \in \Delta^{\mathcal{I}(T,L)}$ and $0 \leqslant i \leqslant N$, $\langle v, vi\rangle \in r^{\mathcal{I}(T,L)}$ if and only if $\mathrm{in}_r \in L(vi)$,
  - for $v \in \Delta^{\mathcal{I}(T,L)}$, $\langle v, v\rangle \in r^{\mathcal{I}(T,L)}$ if and only if $\mathrm{self}_r \in L(v)$,
  - for $v \in \Delta^{\mathcal{I}(T,L)}$ and i such that $vi0^k \in \Delta^{\mathcal{I}(T,L)}$ with $\mathrm{fin}, \mathrm{in}_r \in L(vi0^k)$ for some k, we have $\langle vi0^{k-1}, v\rangle \in r^{\mathcal{I}(T,L)}$.

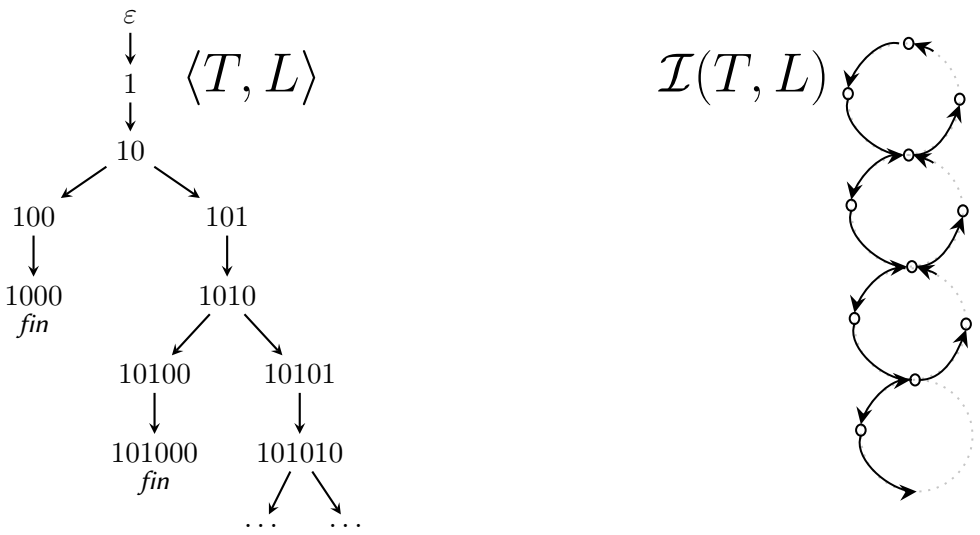
The above construction can be understood as follows. The domain of $\mathcal{I}(T, L)$ consists of all vertices except those labelled with fin, which serve as auxiliary elements used to mark the ends of cycle-guaranteeing paths and to record their final role. The interpretation of concept names is directly inherited from the labelling. The interpretation of roles is defined in three steps. First, r-edges are added between a parent u and its child v whenever $\mathrm{in}_r \in L(v)$. Second, r-self-loops are added at vertices labelled with self_r. Third, cycle-guaranteeing paths of the form $v, vi, vi0, \dots, vi0^k$, where $vi0^k$ is labelled with fin and in_r, are "closed" by adding an r-edge from $vi0^{k-1}$ back to v, effectively "identifying" $vi0^k$ with v. Thus, $\mathcal{I}(T, L)$ can be viewed as a "tree of cycles": traversing a cycle-guaranteeing path returns to the originating vertex, yielding (possibly infinite) chains of cycles that can be navigated both downward and upward. In what follows, encoding-ready trees serve as inputs to tree automata whose nonemptiness is checked by our algorithm. The fin label acts as a signal for the automaton to move from $vi0^k$ back to v. To facilitate this behavior, it is convenient to rely on a two-way variant of tree automata.
We next define tree representations of finite models of $\langle \mathcal{T}, \mathrm{A}\rangle$, in analogy to the so-called Hintikka trees.

Definition 3.2. *An encoding-ready tree $\langle T, L\rangle$ is a* representation *of $\langle \mathcal{T}, \mathrm{A}\rangle$ if the following hold:*

1. *Every vertex not labelled with* fin *satisfies all Boolean constraints from $\mathcal{T}$, and $\mathrm{A} \in L(\varepsilon)$.*
2. *For every i, if C_i is of the form $\exists \mathcal{A}_{q,q'}.\top$, then each vertex v with $\mathrm{A}_i \in L(v)$ either*
   a) *admits a path $v, vi, vi0, \ldots, vi0^k$ such that $vi0^k$ has no 0-child, and assuming $\mathrm{in}_{r_j} \in L(vi0^j)$, the walk $v, r_0, vi, r_1, vi0, \ldots, r_k, vi0^k$ is an $\mathcal{A}_{q,q'}$-witness for v in $\mathcal{I}(T, L)$, or*
   b) *has an $\mathcal{A}_{q,q'}$-witness consisting of the singleton sequence v.*

   Moreover, if $\mathrm{A}_i \notin L(v)$, then there is no $\mathcal{A}_{q,q'}$-witness for v.
3. *For every i, if $\mathrm{C}_i = \exists r.\mathsf{Self}$, then $\mathrm{A}_i \in L(v)$ if and only if $\mathrm{self}_r \in L(v)$.*
4. *For every i, if $\mathrm{C}_i = \exists r^+.\mathsf{Self}$, then $\mathrm{A}_i \in L(v)$ if and only if one of the following holds:*
   a) *there exists a path $v, vi, vi0, \ldots, vi0^k$ such that the final vertex $vi0^k$ is labelled with fin and the walk $v, r, vi, r, vi0, \ldots, vi0^{k-1}, r, v$ is an r^+-self-witness for v in $\mathcal{I}(T, L)$, or*
   b) $\mathrm{self}_r \in L(v)$.
5. Wrappability. *For every infinite path $v_1, v_2, \ldots$, if from some point onward all* in *labels use the same role r, and $\mathrm{C}_i = \exists r^+.\mathsf{Self}$ for some i, then there exists j such that $\mathrm{A}_i \in L(v_k)$ for all $k \geqslant j$.* ◀

Observe that all cycles in this construction are either explicitly required by a vertex v, with the final vertex $vi0^k$ "identified" with v, or arise as combinations of such cycles. In the latter case, a combination may involve multiple role names, and thus cannot violate any negative cycle constraints, or it may involve a single role r, in which case every vertex on the cycle satisfies $\exists r^+.\mathsf{Self}$. Hence, no unintended cycles are introduced by the "two-way nature" of our construction. Nevertheless, not all walks in such a tree are "downward" walks: starting from a vertex somewhere deep within a chain of cycles, one can successively "close" cycles by moving slightly downward and then jumping upward (via the "identification" of $vi0^k$ with v). Subsequently, the walk may continue downward again.



The above picture depicts how a chain of cycles originates from a tree representation. One can show:

Proposition 3.3. *One can construct in time polynomial in $|\mathcal{T}|$ a two-way alternating parity tree automaton $\mathcal{A}_{\mathrm{tree}}$ over the alphabet Σ with polynomially-many states w.r.t. $|\mathcal{T}|$ that accepts exactly the representations of $\langle \mathcal{T}, \mathrm{A}\rangle$. Testing for non-emptiness of its language can be done in time exponential in $|\mathcal{T}|$.* ◀

Proof idea. Such an automaton can be constructed in a standard way, as in many extensions of PDL and the μ-calculus. While routine, the construction is technically tedious; see [21, Secs. 4.2.1 & 4.2.3] for a very detailed exposition. The semantic constraints from Definition 3.2 reduce to checking the existence of paths whose in labels form words in given regular languages, possibly with additional checks on

concept labels along the path. The two-way capability allows navigation along cycle-guaranteeing paths, including upward moves in the tree, while the wrappability condition is handled via the automaton's ability to express fixed-point properties. Full details will be given in the thesis of the second author. The second part of the claim follows from the original work of Vardi [22]. □

As noted at the beginning of this section, our approach to finite satisfiability is automata-based. A pseudocode description is given below; its soundness and completeness are established in next sections.

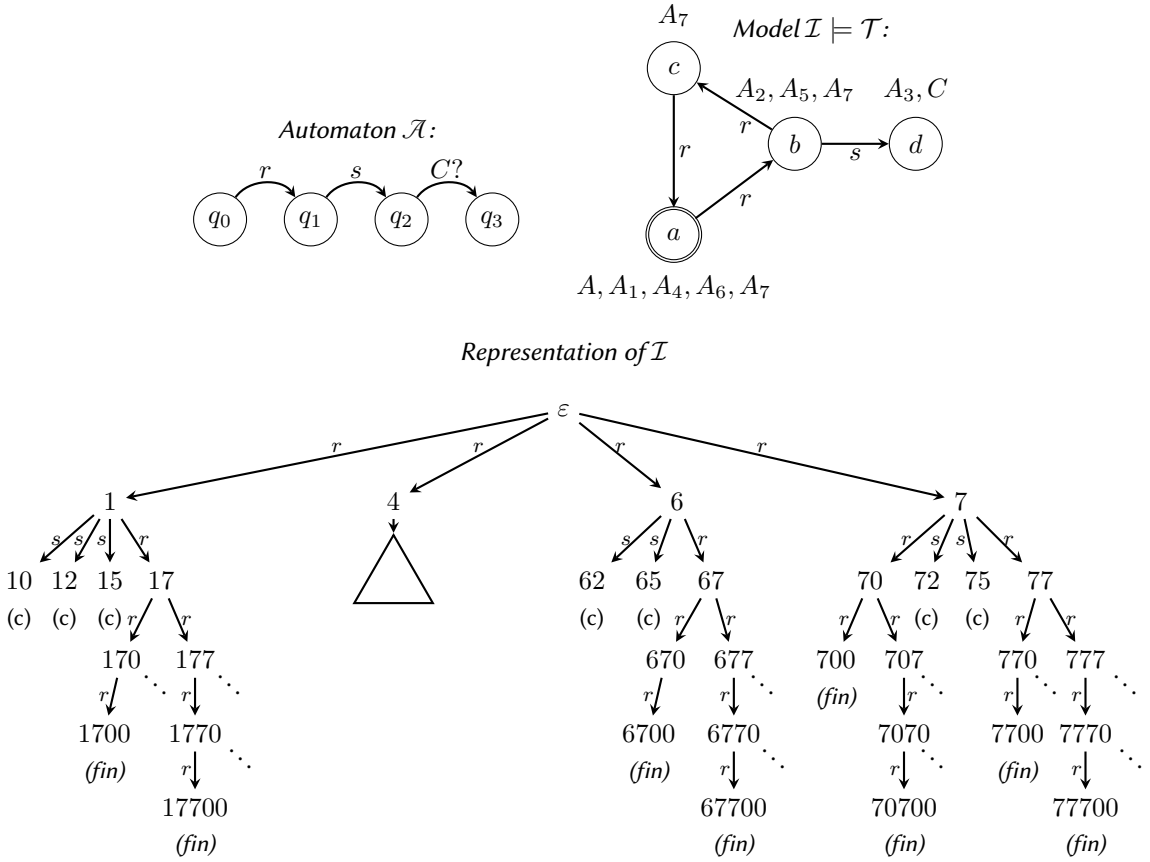
Algorithm 1 Finite satisfiability check for TBoxes in our fragment of LoopPDL.

1: Read the input instance and transform it to normal form, obtaining $\langle \mathcal{T}, \mathrm{A} \rangle$.
2: Construct an automaton $\mathcal{A}_{\mathrm{tree}}$ from Proposition 3.3 that accepts tree representations for $\langle \mathcal{T}, \mathrm{A} \rangle$.
3: Test $\mathcal{A}_{\mathrm{tree}}$ for nonemptiness and accept if its language is nonempty, reject otherwise.

4. Completeness

To prove that our algorithm always detects a finite model whenever one exists, we consider a finite model $\mathcal{I}$ of $\langle \mathcal{T}, \mathrm{A} \rangle$. We construct a representation $\langle T, L \rangle$ accepted by $\mathcal{A}_{\mathrm{tree}}$ via a *TBox-driven unravelling*.

Example 4.1. *Consider a TBox $\mathcal{T}$ in normal form, containing the following concept equivalences:* $\mathrm{A} \equiv \mathrm{A}_1 \sqcap \mathrm{A}_7$, $\mathrm{A}_1 \equiv \exists \mathcal{A}_{q_0,q_3}.\top$, $\mathrm{A}_2 \equiv \exists \mathcal{A}_{q_1,q_3}.\top$, $\mathrm{A}_3 \equiv \exists \mathcal{A}_{q_2,q_3}.\top$, $\mathrm{A}_4 \equiv \exists \mathcal{A}_{q_0,q_2}.\top$, $\mathrm{A}_5 \equiv \exists \mathcal{A}_{q_1,q_2}.\top$, $\mathrm{A}_6 \equiv \exists \mathcal{A}_{q_0,q_1}.\top$, $\mathrm{A}_7 \equiv \exists r^+.\mathsf{Self}$, $\mathrm{C} \equiv \neg \mathrm{A}_7$ *(some NFAs with empty languages or accepting only singleton walks are omitted), where the automaton $\mathcal{A}$ and a model $\mathcal{I}$ of $\mathcal{T}$ are depicted below.*



Lemma 4.2. *If there exists a finite model of $\langle \mathcal{T}, \mathrm{A} \rangle$, then the automaton $\mathcal{A}_{\mathrm{tree}}$ accepts some tree.* ◀

Proof. Let $a \in \Delta^{\mathcal{I}}$ be an element at which A is satisfied. Starting from a, which becomes the root, we successively add all necessary witnesses—arranged in (cycle-)guaranteeing paths—to vertices whose demands have not yet been satisfied. Wrappability follows from the fact that every finite fragment of a path in our unravelling corresponds to a walk in $\mathcal{I}$, and walks of sufficient length in finite interpretations eventually "get stuck" on some cycle. In the unravelling process, we focus on the TBox $\mathcal{T}$ in normal form as fixed at the end of Section 2 and append witnesses according to the demands it determines. For each element $b \in \Delta^{\mathcal{I}}$, we select its witnessing walks and cycles in advance. If C_i is a walk demand and $b \in \mathrm{C}_i^{\mathcal{I}}$, let $W_i(b)$ denote a walk in $\mathcal{I}$ that starts at b and witnesses C_i. If C_i is a cycle demand, we let $W_i(b)$ be an appropriate cycle. While building the labelled tree $\langle T, L\rangle$, we maintain two auxiliary objects: the *projection* function $\Pi\colon \Delta^{\mathcal{I}(T,L)} \to \Delta^{\mathcal{I}}$ and the set U_j of *pending vertices* whose demands are not yet satisfied at the j-th stage of the construction. As one might expect, the former is only extended at each stage, whereas the latter changes completely from one stage to the next.

We begin with a tree T containing just the root, $U_0 = \{\varepsilon\}$, and $\Pi(\varepsilon) = a$. We set $L(\varepsilon)$ to contain the concept names satisfied at a, together with self_r for every r such that $\langle a, a\rangle \in r^{\mathcal{I}}$. Now suppose U_j has been constructed. For every demand C_i and every vertex $v \in U_j$, we proceed as follows.

- If C_i is a walk demand and $\Pi(v) \in \mathrm{C}_i^{\mathcal{I}}$, let $W_i(\Pi(v)) \coloneqq a_0, r_1, a_1, \dots, r_k, a_k$ be the associated witnessing path. If $k = 0$, we do nothing. Otherwise, we add the vertices $vi, vi0, \dots, vi0^{k-1}$ to T and to U_{j+1}.
- If C_i is a cycle demand and $\Pi(v) \in \mathrm{C}_i^{\mathcal{I}}$, let $W_i(\Pi(v)) \coloneqq a_0, r, a_1, \dots, r, a_k, r, a_0$ be the associated witnessing cycle. If it is a self-loop, we do nothing. Otherwise, we add the vertices $vi, vi0, \dots, vi0^k$ to T and include all of them except $vi0^k$ in U_{j+1}. We set $\mathrm{fin}, \mathrm{in}_r \in L(vi0^k)$.

In both cases, for $0 \leqslant \ell \leqslant k-1$, we let $L(vi0^\ell)$ contain the concept names satisfied at $a_{\ell+1}$, the label $\mathrm{in}_{r_{\ell+1}}$, and self_r for every r such that $a_{\ell+1}$ has an r-self-loop in $\mathcal{I}$. Finally, we set $\Pi(vi0^\ell) = a_{\ell+1}$.

The resulting tree T consists of all vertices added at some stage of the above process. The label and projection of each vertex are fixed at the moment of its inclusion. It remains to show that $\langle T, L\rangle$ is indeed a representation of $\langle \mathcal{T}, \mathrm{A}\rangle$. Clearly, $\mathrm{A} \in L(\varepsilon)$, and every vertex satisfies the Boolean constraints in $\mathcal{T}$, since the set of concepts satisfied at any vertex is inherited from one in $\mathcal{I}$. Thus, Condition 1 of Definition 3.2 holds. The existence of (cycle-)guaranteeing paths and required self-loops, as well as the absence of forbidden self-loops follows by construction. It remains to show that no forbidden walks or cycles exist in $\mathcal{I}(T, L)$. For cycles, the argument is straightforward: a vertex can only reach itself by traversing a witnessing cycle on which it lies, possibly passing through other witnessing cycles along the way. Hence, either all roles appearing on such a cycle coincide—in which case the cycle has the same shape as the witnessing cycle—or at least two distinct roles occur on it, and the cycle therefore cannot be forbidden, since only cycles of the form r^+ or r can be demanded or forbidden. In particular, reaching oneself via an r^+-cycle requires lying on a cycle-guaranteeing path for $\exists r^+.\mathsf{Self}$. As for walks, observe the following property of Π: for every role name r and every edge $\langle v, w\rangle \in r^{\mathcal{I}(T,L)}$, it holds by construction that $\langle \Pi(v), \Pi(w)\rangle \in r^{\mathcal{I}}$; that is, Π is a homomorphism from $\mathcal{I}(T, L)$ to $\mathcal{I}$ (and, just as plainly, Π preserves the satisfaction of concept names). Consequently, every walk π in $\mathcal{I}(T, L)$ projects via Π to a walk in $\mathcal{I}$ that witnesses the same demands and has the same set of concepts satisfied at its initial element.

It remains to verify the wrappability condition. Consider a role name r such that $\mathrm{C}_i = \exists r^+.\mathsf{Self}$ for some i, and let $v_1, v_2, \dots$ be an infinite path in T such that $\mathrm{in}_r \in L(v_j)$ for all $j \geqslant k_0$, where k_0 is some fixed index. Define $a_j \coloneqq \Pi(v_j)$. The sequence $a_1, a_2, \dots$ is infinite but takes only finitely many values due to the finiteness of the domain, so there exists $k_1 \geqslant k_0$ such that for every $j \geqslant k_1$ the element a_j occurs infinitely often in the sequence. Take any $j \geqslant k_1$; we claim that $\mathrm{A}_i \in L(v_j)$. Since a_j occurs infinitely often, let a_ℓ denote its next occurrence after position j. As established above, Π is a homomorphism, so $a_j, r, a_{j+1}, r, \dots, r, a_\ell$ is a walk in $\mathcal{I}$; and since $a_j = a_\ell$, this walk is in fact a cycle. Because $\mathcal{I} \models \mathcal{T}$, it follows that $a_j \in \mathrm{A}_i^{\mathcal{I}}$, and hence by construction $\mathrm{A}_i \in L(v_j)$. □

5. Soundness

Suppose that $\mathcal{A}_{\text{tree}}$ from Algorithm 1 accepts a labelled tree $\langle T_{\text{acc}}, L\rangle$. By Rabin's Basis Theorem [23, Thm. 3.7, p. 424], we may assume without loss of generality that $\langle T_{\text{acc}}, L\rangle$ is regular. Our next goal is to show how to transform $\langle T_{\text{acc}}, L\rangle$ into a finite model of $\langle \mathcal{T}, \mathrm{A}\rangle$ by wrapping it into a finite structure, with an iterative application of the wrapping operation (formally defined in Definition 5.5). To handle intermediate interpretations arising during the wrapping process of tree representations (as well as to formalize the invariants required for correctness) we introduce *wrappable interpretations*.

Definition 5.1. *A* wrappable interpretation $\mathcal{W}$ *is a tuple* $\langle \mathcal{I}, E, S, \lambda\rangle$ *satisfying:*

1. $\Delta^{\mathcal{I}}$ *is a regular tree of finite degree, and all* 0*-paths in* $\Delta^{\mathcal{I}}$ *are finite.*
2. $E, S\colon \mathsf{rol}(\mathcal{T}) \to \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ *assign to each role name* r *a subset of* $r^{\mathcal{I}}$*. For any role name* r*, the pairs contained in* $E(r)$ *are called* plain *edges, whereas those contained in* $S(r)$ *are called* special *edges.*
3. λ *maps nodes of* $\Delta^{\mathcal{I}}$ *to subsets of* $\mathsf{con}(\mathcal{T})$ *(called* types*). We write* $v \triangleright \mathrm{C}_i$ *if* $\mathrm{A}_i \in \lambda(v)$ *for* $(\mathrm{A}_i \equiv \mathrm{C}_i) \in \mathcal{T}$*, and* $v \not\triangleright \mathrm{C}_i$ *otherwise (recall the framed text in Section 2).*
4. *For every child* vi *of* v*, there exists a role name* r *such that* $\langle v, vi\rangle \in E(r)$*.*
5. *(*Integrity*) If* $\mathrm{A}_i \in \lambda(v)$ *and* v *is not its own witness (cf. Def. 3.2, 2b, 4b), then* v *has a witnessing walk of the form* $v, r_0, wi, r_1, wi0, \dots, r_k, wi0^k$ *if* C_i *is a walk demand, or* $v, r, wi, r, wi0, \dots, r, wi0^{k-1}, r, v$ *if* C_i *is a cycle demand (mind the* w*!). Moreover, if* $w \neq v$*, then* v *is a descendant of* wi*.*
6. *(*Wrappability*, cf. Def. 3.2, 5). For every infinite path* $v_1, v_2, \dots$ *in* $\Delta^{\mathcal{I}}$ *and every* $r \in \mathbf{N}_{\mathbf{R}}^{\circlearrowleft}$*, if there exists* i *such that* $\langle v_k, v_{k+1}\rangle \in E(r)$ *for all* $k \geqslant i$*, then there exists* j *such that* $v_k \triangleright \exists r^+.\mathsf{Self}$ *for all* $k \geqslant j$*.* ◀

Note that the interpretation $\mathcal{I}_0 \coloneqq \mathcal{I}(T_{\text{acc}}, L)$ induced by the tree representation $\langle T_{\text{acc}}, L\rangle$ can be lifted to a wrappable interpretation $\mathcal{W}_0 \coloneqq \langle \mathcal{I}_0, E_0, S_0, \lambda_0\rangle$ as follows: (i) each $E_0(r)$ contains all parent-to-child r-connections, while all remaining r-edges belong to $S_0(r)$, and (ii) for each $v \in \Delta^{\mathcal{I}_0}$, the type $\lambda_0(v)$ consists of all concept names satisfied by v in $\mathcal{I}_0$. Thus, special edges in $\mathcal{W}_0$ are precisely those which close cycles. More generally, plain edges of wrappable interpretations correspond to the tree structure and may later be redirected during the wrapping process, whereas special edges serve to close cycles demanded by elements and may also be modified. Condition 5 of Definition 5.1 ensures that witnessing walks are not redirected midway.

In our construction, a vital role is played by the notion of a *trace*. The trace of an element captures its position relative to elements that forbid self-reachability, allowing us to avoid introducing forbidden cycles during the finitization process. Given a wrappable interpretation $\mathcal{W} \coloneqq \langle \mathcal{I}, E, S, \lambda\rangle$ and a vertex $v \in \Delta^{\mathcal{I}}$, let $\wr(v)$ denote the unique root-to-v walk in $\mathcal{I}$ (consisting only of edges of the form $\langle w, wi\rangle$). For a role name r, let $\wr_r(v)$ be the longest suffix of $\wr(v)$ containing only edges labelled with r. We have:

Definition 5.2. *Let* $r \in \mathbf{N}_{\mathbf{R}}^{\circlearrowleft}$ *and let* $\mathcal{W} \coloneqq \langle \mathcal{I}, E, S, \lambda\rangle$ *be a wrappable interpretation. The* trace *function* $\mathrm{tr}_r\colon \Delta^{\mathcal{I}} \to \mathbb{N}$ *is defined by* $\mathrm{tr}_r(v) \coloneqq \big|\{\, w \mid w \text{ occurs in } \wr_r(v) \text{ and } w \not\triangleright \exists r^+.\mathsf{Self} \,\}\big|$*.* ◀

A crucial invariant of the construction is that trace values remain unchanged under modifications of the interpretation. Hence, traces determined for the initial interpretation induced by $\langle T_{\text{acc}}, L\rangle$ can be used throughout the construction. Moreover, traces take only finitely many values as proven below.

Lemma 5.3. *For every* $r \in \mathbf{N}_{\mathbf{R}}^{\circlearrowleft}$ *and wrappable interpretation* $\mathcal{W} \coloneqq \langle \mathcal{I}, E, S, \lambda\rangle$*, the image of* tr_r *is finite.* ◀

Proof. Ad absurdum, suppose that tr_r takes infinitely many values. Since the tree underlying $\mathcal{I}$ is regular, there exists a vertex $v \in \Delta^{\mathcal{I}}$ such that $\mathrm{tr}_r(v)$ exceeds the number of distinct subtrees of $\mathcal{I}$. Consider the walk $\wr(v)$. By assumption, it contains more vertices w with $w \not\triangleright \exists r^+.\mathsf{Self}$ than there are distinct subtree types. Hence, there exist two such vertices w_1 and w_2 on $\wr(v)$ whose subtrees are isomorphic. By construction, there is a walk from w_1 to w_2 consisting only of r-edges. Now, following the path from the root to w_1 and then iterating the segment isomorphic to the one from w_1 to w_2 yields an infinite path violating the wrappability condition. A contradiction. □

To make the construction precise, we formalize its key invariants. Throughout the remainder of this section, we call $\mathcal{W} \coloneqq \langle \mathcal{I}, E, S, \lambda \rangle$ *good* if $\varepsilon \in \mathrm{A}^{\mathcal{I}}$, $\mathcal{I}$ is a model of $\mathcal{T}$, and for all $r \in \mathbf{N}_{\mathbf{R}}^{\circlearrowleft}$ and $v, w \in \Delta^{\mathcal{I}}$:

1. if $\langle v, w \rangle \in E(r)$, then $\mathrm{tr}_r(v) \leqslant \mathrm{tr}_r(w)$, with strict inequality if $w \not\triangleright \exists r^+.\mathsf{Self}$,
2. if $\langle v, w \rangle \in S(r)$, then $\mathrm{tr}_r(v) = \mathrm{tr}_r(w)$,
3. if $\langle v, w \rangle \in S(r)$, then $w \triangleright \exists r^+.\mathsf{Self}$.

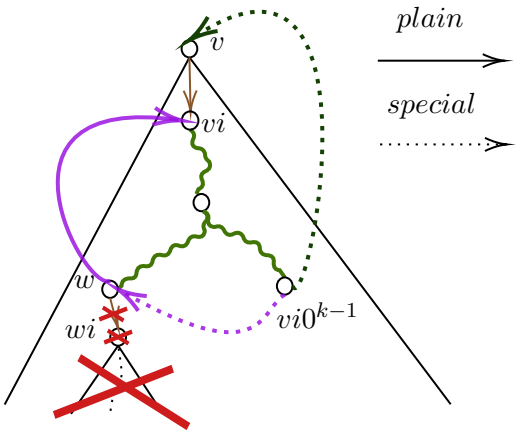
Hence, traces do not decrease along plain r-edges and remain constant along special r-edges. We have:

Lemma 5.4. *The interpretation $\mathcal{W}_0$ induced by $\langle T_{\mathrm{acc}}, L \rangle$ is good.* ◀

Proof. By Condition 1 of Definition 3.2, we have $\mathrm{A} \in \lambda(\varepsilon)$ and $\mathcal{I}_0$ satisfies all Boolean constraints of $\mathcal{T}$. Satisfaction of walk demands follows from Item 2, while cycle demands are ensured by Items 3 and 4 of Definition 3.2. Condition 1 is immediate from the definition of $\wr_r(\cdot)$. Extending a walk π ending at v by an r-edge to a walk π, r, vi yields $\wr_r(v) \subseteq \wr_r(vi)$, so the number of elements forbidding an r^+-cycle does not decrease, and increases if $vi \not\triangleright \exists r^+.\mathsf{Self}$. Conditions 2 and 3 hold since any vertex receiving a special r-edge has an r^+-witnessing cycle and thus demands an r^+-cycle. □

We also define *similarity*. Vertices v, w of $\Delta^{\mathcal{I}}$ are *similar* (denoted $v \approx w$) if $\lambda(v) = \lambda(w)$ and $\mathrm{tr}_r(v) = \mathrm{tr}_r(w)$ for all $r \in \mathbf{N}_{\mathbf{R}}^{\circlearrowleft}$. Plain edges $\langle v_1, w_1 \rangle$ and $\langle v_2, w_2 \rangle$ are *similar* (denoted $\langle v_1, w_1 \rangle \approx \langle v_2, w_2 \rangle$) if their endpoints are similar, there is an $i \neq 0$ such that both w_1 and w_2 are the i-th children of their parents, and there exists r such that both of these pairs belong to $E(r)$.

The following operation constitutes a single step in the process of finitizing the structure. Its intuition is as follows. We consider two similar edges $\langle v, vi \rangle$ and $\langle w, wi \rangle$ such that w is a descendant of vi. We then redirect the edge $\langle w, wi \rangle$ to vi and remove the subtree rooted at wi (including wi). Moreover, if wi guaranteed an r-cycle for w, we restore an r-cycle for w by reusing the r-cycle guaranteed for v, after adding a *special* r-edge from the last vertex on a cycle-guaranteeing path. This is illustrated below.



Definition 5.5. *Let $\mathcal{W} \coloneqq \langle \mathcal{I}, E, S, \lambda \rangle$ be a wrappable interpretation and let $1 \leqslant i \leqslant N$. Fix similar edges $\langle v, vi \rangle \approx \langle w, wi \rangle$ such that w is a descendant of vi. A* wrapping operation *$\langle \langle v, vi \rangle, \langle w, wi \rangle \rangle$ performed on $\mathcal{W}$ results in the wrappable interpretation $\mathcal{W}' = \langle \mathcal{I}', E', S', \lambda' \rangle$ in which all the components are as in $\mathcal{W}$ restricted to $\Delta^{\mathcal{I}}$ with wi and its descendants removed, modulo the following modifications:*

- *$E'(s)$ and $s^{\mathcal{I}'}$ contain $\langle w, vi \rangle$, for the unique role name s with $\langle v, vi \rangle, \langle w, wi \rangle \in E(s)$, and*
- *Suppose $\mathrm{C}_i = \exists s^+.\mathsf{Self}$ and consider a witnessing walk $v, s, vi, s, vi0, \ldots, vi0^{k-1}, s, v$ for this cycle demand and v (see the integrity condition). Then both $S'(s)$ and $s^{\mathcal{I}'}$ contain $\langle vi0^{k-1}, w \rangle$.* ◀

It is immediate that $\mathcal{W}'$ is wrappable. The only subtle point is the preservation of integrity (Item 5 of Definition 5.1). This is addressed in the appendix B.1.

The following lemma shows that the wrapping operation preserves the property of being "good" for wrappable interpretations, and is central to the soundness of our algorithm. Its key idea is the use of traces to ensure that no forbidden cycles are introduced.

Lemma 5.6. *Let the wrapping operation $\langle \langle v, vi \rangle, \langle w, wi \rangle \rangle$ be performed on $\mathcal{W}$, resulting in $\mathcal{W}'$, as in Definition 5.5, and assume that $\mathcal{W}$ is good. Then $\mathcal{W}'$ is good.* ◀

Proof sketch. The verification of all conditions except the satisfaction of GCIs of the form $\mathrm{A}_n \equiv \exists r^+.\mathsf{Self}$ is routine and deferred to the appendix. By construction, all elements (possibly except w) satisfying A_n retain their r^+-self-witnesses, while w, if necessary, can reuse such a witness from v. It thus remains to show that no forbidden r^+-cycles are introduced. First observe that for every r-walk $u_0, r, u_1, \ldots, r, u_k$, we have $\mathrm{tr}_r(u_j) \leqslant \mathrm{tr}_r(u_\ell)$ for all $j \leqslant \ell$. This follows from Conditions 1 and 2 of the goodness property by straightforward induction on k. Now let u be a vertex satisfying $\exists r^+.\mathsf{Self}$ in $\mathcal{I}'$. Then there exists an r-cycle $u, r, u_1, \ldots, r, u_{k-1}, r, u$. If $k = 1$, this is a self-loop, which is already present in $\mathcal{I}$. Since $\mathcal{I} \models \mathcal{T}$ and the construction of $\mathcal{I}'$ preserves types, u satisfies A_n. Assume $k > 1$ and, towards a contradiction, that $u \not\models \exists r^+.\mathsf{Self}$. By Cond. 3, the edge $\langle u_{k-1}, u\rangle$ is plain. Then, by Cond. 1, $\mathrm{tr}_r(u_{k-1}) < \mathrm{tr}_r(u)$, while by the above observation, $\mathrm{tr}_r(u) \leqslant \mathrm{tr}_r(u_{k-1})$. A contradiction. □

Since by Lemma 5.4 we know that $\mathcal{W}_0$ (i.e., our initial wrappable interpretation) is good, it remains to devise a systematic procedure for applying wrapping operations that eventually yields a finite wrappable interpretation. By the goodness property, the resulting interpretation is going to be a finite model of $\langle \mathcal{T}, \mathrm{A}\rangle$. As an auxiliary lemma, we show that every infinite path contains a pair of sufficiently distant similar edges. The proof is by pigeonhole principle (see the appendix).

Lemma 5.7. *In any wrappable interpretation $\mathcal{W} \coloneqq \langle \mathcal{I}, E, S, \lambda\rangle$, on every infinite path $v_1, v_2, \ldots$ there are some vertices $v_i, v_{i+1}, v_j, v_{j+1}$ such that $i + 2 \leqslant j$ and $\langle v_i, v_{i+1}\rangle \approx \langle v_j, v_{j+1}\rangle$.* ◀

We are finally ready to establish the main result of this section.

Lemma 5.8. *If $\mathcal{A}_{\mathrm{tree}}$ accepts $\langle T_{\mathrm{acc}}, L\rangle$, then there exists a finite model of $\langle \mathcal{T}, \mathrm{A}\rangle$.* ◀

Proof. Let the *depth* of a node v be $|v|$, i.e., the length of its underlying sequence. We define the inductive "wrapping" process as follows. Suppose $\mathcal{W}_n$ is given, and let D be the set of all vertices in $\mathcal{W}_n$ of depth $n{+}1$ of the form $viwi$, where $\langle v, vi\rangle \approx \langle viw, viwi\rangle$ for some $|w| \geqslant 1$. Define $\mathcal{W}_{n+1}$ as the result of applying the wrapping operations $\langle\langle v, vi\rangle, \langle viw, viwi\rangle\rangle$ for all $viwi \in D$, in arbitrary order. To prove the existence of a finite model, it suffices to prove existence of some m for which $\mathcal{W}_m$ contains no infinite paths. Towards a contradiction, suppose that for every m, $\mathcal{W}_m$ contains a vertex of depth m. Define $X \coloneqq \{v \in T_0 \mid \forall n \geqslant |v|\ \exists w \text{ with } |vw| = n \text{ and } vw \in T_n\}$. Thus, X consists of vertices in $\mathcal{W}_0$ that, for every n, have in their subtree a vertex at depth n surviving the n-th "stage" of the wrapping process. We use the following observations (proved in the appendix): (1) $\varepsilon \in X$, (2) if $v \in X$, then $vi \in X$ for some i, and (3) if $vi \in X$, then the path from the root to v contains no edge similar to $\langle v, vi\rangle$ except possibly its immediate predecessor. These imply the existence of an infinite path in T_0 with no pair of similar edges separated by at least one edge, contradicting Lemma 5.7. □

6. Conclusions

We introduced a new automata-based approach to the finite satisfiability problem for PDL extended with the generalized Self operator. While the problem in full generality remains open for over 40 years, we resolved it for the fragment of LoopPDL where Self is applied only to atomic roles or their transitive closures. Since the presented Algorithm 1 is sound and complete, we can conclude:

Theorem 6.1. *The finite satisfiability problem for* LoopPDL*-TBoxes in which the* Self *operator is restricted to atomic roles or their transitive closures is decidable and ExpTime-complete.* ◀

Our approach represents finite models as infinite, automata-recognisable trees satisfying a wrappability condition that enables their transformation into finite models. While completeness extends readily to full LoopPDL, establishing soundness is more challenging. By refining the wrappability condition and adapting the automaton construction from Algorithm 1, we can already handle more expressive fragments. We conjecture that the method extends to full LoopPDL, although a general correctness proof remains open (but we are intensively working on it). As future work, we plan to develop similar techniques to finite-model reasoning in other logics with finite satisfiability problems still open. This forms a part of our recently started FWF project titled "FINGO: Finite Graph Operating Automata Meet Dynamic Logics". Drop us a line if you are interested in collaborating on the project!

Acknowledgments

This work was supported by NCN grants 2024/53/N/ST6/01390 (B. Bednarczyk) and 2021/41/B/ST6/00996 (M. Swoboda). This work is a part of a forthcoming MSc thesis of M. Swoboda, supervised by B. Bednarczyk at the University of Wrocław.

Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT and Claude in order to: grammar and spelling check, polish the text. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] M. Fischer, R. Ladner, Propositional Dynamic Logic of Regular Programs, Journal of Computer and System Sciences (1979).
[2] D. Harel, J. Tiuryn, D. Kozen, Dynamic Logic, MIT Press, 2000.
[3] N. Troquard, P. Balbiani, Propositional Dynamic Logic, in: The Stanford Encyclopedia of Philosophy, 2023.
[4] V. Pratt, A Near-Optimal Method for Reasoning about Action, Journal of Computer and System Sciences (1980).
[5] D. Calvanese, M. Ortiz, M. Šimkus, Verification of Evolving Graph-structured Data under Expressive Path Constraints, in: Proceedings of International Conference on Database Theory (ICDT), 2016.
[6] B. Bednarczyk, E. Kieroński, Finite Entailment of Local Queries in the Z Family of Description Logics, in: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 2022.
[7] R. Danecki, Propositional Dynamic Logic with Strong Loop Predicate, in: Proceedings of Mathematical Foundations of Computer Science (MFCS), 1984.
[8] D. Harel, Recurring Dominoes: Making the Highly Undecidable Highly Understandable (Preliminary Report), in: Proceedings of Foundations of Computation Theory (FCT), 1983.
[9] M. Lange, A Lower Complexity Bound for Propositional Dynamic Logic with Intersection, in: Proceedings of Advances in Modal Logic (AIML), 2004.
[10] M. Lange, C. Lutz, 2-ExpTime Lower Bounds for Propositional Dynamic Logics with Intersection, Journal of Symbolic Logic (2005).
[11] C. Lutz, PDL with Intersection and Converse Is Decidable, in: Proceedings of Computer Science Logic (CSL), 2005.
[12] S. Göller, M. Lohrey, C. Lutz, PDL with Intersection and Converse Is 2EXP-Complete, in: Proceedings of Foundations of Software Science and Computational Structures (FOSSACS), 2007.
[13] S. Göller, M. Lohrey, C. Lutz, PDL with Intersection and Converse: Satisfiability and Infinite-state Model Checking, Journal of Symbolic Logic (2009).
[14] Y. Nakamura, Guarded Negation Transitive Closure Logic is 2-EXPTIME-complete, ArXiv (2025).
[15] S. Göller, Computational Complexity of Propositional Dynamic Logics, Ph.D. thesis, University of Leipzig, 2008.
[16] D. Figueira, S. Figueira, E. Pin Baque, PDL on Steroids: on Expressive Extensions of PDL with Intersection and Converse, in: Proceedings of Logic in Computer Science (LICS), 2023.
[17] B. Bednarczyk, E. Kieroński, Guarded Fragments Meet Dynamic Logic: The Story of Regular Guards, in: Proceedings of Principles of Knowledge Representation and Reasoning (KR), 2025.
[18] E. Kieroński, L. Tendera, Finite Satisfiability of the Two-Variable Guarded Fragment with Transitive Guards and Related Variants, ACM Transactions on Computational Logic (2018).
[19] V. Gutiérrez-Basulto, A. Gutowski, Y. Ibáñez-García, F. Murlak, Finite Entailment of UCRPQs over ALC Ontologies, in: Proceedings of Principles of Knowledge Representation and Reasoning (KR), 2022.

[20] A. Gutowski, Regular Path Queries and Modal Constraints: Decidability of Query Containment under a Graph Database Schema, Ph.D. thesis, University of Warsaw, 2026.
[21] D. Calvanese, T. Eiter, M. Ortiz, Answering Regular Path Queries in Expressive Description Logics via Alternating Tree-Automata, Information and Computation (2014).
[22] M. Y. Vardi, Reasoning about the Past with Two-way Automata, in: Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP), 1998.
[23] D. Perrin, J. Éric Pin, Infinite Trees, in: Infinite Words, 2004.

A. Appendix: Preliminaries

A.1. Proof of Lemma 2.1

Proof. We describe a polynomial-time procedure that transforms an arbitrary TBox $\mathcal{T}$ into an equi-satisfiable TBox in normal form. The construction proceeds in several stages, each of which preserves equi-satisfiability and increases the size of the TBox by at most a polynomial factor.

Let $\mathcal{T}$ be an arbitrary input TBox. We first transform $\mathcal{T}$ so that every regular expression occurring in it mentions only concept names rather than arbitrary complex concepts. To this end, pick any regular expression in $\mathcal{T}$ that contains a complex concept C, introduce a fresh concept name A_{C}, and extend $\mathcal{T}$ with the axiom $A_{\mathrm{C}} \equiv \mathrm{C}$. We then replace every occurrence of C inside the regular expression by A_{C}. The resulting regular expression is equi-satisfiable with the original under the extended TBox, since A_{C} and C are forced to denote the same set of elements in every model; moreover, it no longer mentions C as a subexpression. Repeating this step exhaustively—which terminates after polynomially many iterations, as each application strictly reduces the number of complex-concept occurrences inside regular expressions—yields a TBox in which every regular expression is built only from concept names, role names, and the regular operators.

We now have a TBox $\mathcal{T}$ in which regular expressions contain no complex concepts, and we proceed to compile these regular expressions into automata. For each regular expression π occurring in $\mathcal{T}$ that is *not* in the scope of a loop operator, we construct, by the standard Thompson construction, an equi-satisfiable nondeterministic finite automaton with a single initial state $q_{0,\pi}$ and a single final state $q_{f,\pi}$. Let $\mathcal{A}$ be the NFA obtained by taking the disjoint union of all these individual automata; thus $\mathcal{A}$ has a distinguished pair of states $(q_{0,\pi}, q_{f,\pi})$ for every such regular expression π. We then replace each occurrence of π in $\mathcal{T}$ by $\mathcal{A}_{q_{0,\pi},\, q_{f,\pi}}$. By construction, $\mathcal{A}_{q_{0,\pi},\, q_{f,\pi}}$ recognizes exactly the language denoted by π, so this rewriting preserves equi-satisfiability. Finally, for each pair of states q and q' of $\mathcal{A}$, we append to $\mathcal{T}$ the axiom $A_{q,q'} \equiv \exists \mathcal{A}_{q,q'}.\top$, where $A_{q,q'}$ is a fresh concept name.

Call a concept *Boolean* if it contains no existential restriction, i.e., if it is built from concept names using only the Boolean connectives. We pick any non-Boolean concept C in $\mathcal{T}$ whose top-level connective is an existential restriction, introduce a fresh concept name A_{C}, and add the axiom $A_{\mathrm{C}} \equiv \mathrm{C}$ to $\mathcal{T}$. We then replace every occurrence of C in $\mathcal{T}$—other than the one in the newly added defining axiom—by A_{C}. The resulting TBox is equi-satisfiable with the previous one, since A_{C} is forced to be interpreted as C, and it contains strictly fewer non-Boolean subconcepts. Iterating this step exhaustively, again for polynomially many rounds, produces a TBox in which every concept is Boolean.

The TBox obtained at the end of this procedure is in normal form and is equi-satisfiable with the original $\mathcal{T}$. Since each of the stages above introduces only polynomially many fresh concept names and axioms and runs in polynomial time, the overall transformation is polynomial. □

B. Appendix: Soundness

B.1. Detailed proof of Lemma 5.6

Proof. Consider $\mathcal{W}$, $\mathcal{W}'$, $\langle v, vi\rangle$ and $\langle w, wi\rangle$ as in the statement of the lemma. Let T_{wi} be the set of all vertices deleted from $\Delta^{\mathcal{I}}$ as a result of the operation (i.e., the subtree of wi). Assume that $\mathcal{W}$ is good. We additionally assume that C_i is an s^+-cycle demand—the other case just requires verifying fewer conditions. Let us show that $\mathcal{W}'$ is good. First, let us see that the operation preserves Conditions 1, 2 and 3. The value of $\wr(u)$ is not affected for any $u \in T'$, since the only change done to edges originating from T is the removal of wi and its descendants, thus all trace values remain the same as in $\mathcal{W}$. The only new edges are $\langle w, vi\rangle \in E'(s)$ and $\langle vi0^{k-1}, w\rangle \in S'(s)$. Call them e_1 and e_2 respectively. It suffices to show that they obey the desired conditions.

Because $e_1 \in E'(s)$, e_1 must obey Condition 1. Since $\mathcal{W}$ is good and $\langle v, vi\rangle \in E(s)$, we have $\mathrm{tr}_s(v) \leqslant \mathrm{tr}_s(vi)$ (strict if $vi \not\Vdash \exists s^+.\mathsf{Self}$). By the fact that $w \approx v$, we conclude $\mathrm{tr}_s(w) = \mathrm{tr}_s(v)$, and hence $\mathrm{tr}_s(w) \leqslant \mathrm{tr}_s(vi)$ (strict if $vi \not\Vdash \exists s^+.\mathsf{Self}$).

Because $e_2 \in S'(s)$, e_2 must obey Conditions 2 and 3. Since $\mathcal{W}$ is good and $\langle vi0^{k-1}, v\rangle \in S(s)$, we have $\mathrm{tr}_s(v) = \mathrm{tr}_s(vi0^{k-1})$. Again, from the equality $\mathrm{tr}_s(w) = \mathrm{tr}_s(v)$ it follows that $\mathrm{tr}_s(w) = \mathrm{tr}_s(vi0^{k-1})$. Now, the witnessing cycle $v, s, vi, s, vi0, \ldots, s, vi0^{k-1}, s, v$ exists in $\mathcal{I}$, and since $\mathcal{I} \models \mathcal{T}$, thus $v \triangleright \exists s^+.\mathsf{Self}$. By the similarity $v \approx w$, we obtain $\lambda(v) = \lambda(w)$, and hence $w \triangleright \exists s^+.\mathsf{Self}$.

Let us now show the remaining conditions. Obviously, $\mathrm{A} \in \lambda'(\varepsilon)$. We concentrate on the TBox $\mathcal{T}$.

Boolean constraints. The Boolean constraints are satisfied because λ' is just a restriction of λ, thus no vertex has its type changed.

Walk-demand constraints. For one direction, consider any $u \in \Delta^{\mathcal{I}'}$ and assume $u \triangleright \exists \mathcal{A}_{q,q'}.\top$. If u satisfies this demand in $\mathcal{W}$ without witnesses, then it does so in $\mathcal{W}'$ as well. Otherwise, by integrity of $\mathcal{W}$, there is a witnessing walk of the form $u'j, r_1, u'j0, \ldots, r_k, u'j0^k$ in $\mathcal{I}$. If it contains no element of T_{wi}, then it is present in $\mathcal{I}'$ as well. Assume, therefore, that this walk contains an element of T_{wi}. Then, $u'j \in T_{wi}$, because $i \neq 0$. Now, by integrity again, if $u \neq u'$, then u is a descendant of $u'j$. If $u \neq u'$ held, then u would belong to T_{wi}, a contradiction. Hence, $u = u'$. Now, because $uj \in T_{wi}$ and $u \in \Delta^{\mathcal{I}'}$, we obtain $uj = wi$. By our initial assumption, C_i is a cycle demand, making this case contradictory. If we considered the case of C_i being a walk demand, it would now suffice to point out the path $w, s, vi, r_1, vi0, \ldots, r_k, vi0^k$, with $\langle w, vi\rangle$ and $\langle v, vi\rangle$ being of the same role and $v \approx w$.

We prove the other direction by induction on the number of occurrences of edges from $\{e_1, e_2\}$ in a walk. Specifically, we claim that for every $n \in \mathbb{N}$ and $u \in \Delta^{\mathcal{I}'}$, if a walk witnessing the fact that $u \in (\exists \mathcal{A}_{q,q'}.\top)^{\mathcal{I}'}$ has n occurrences of edges from $\{e_1, e_2\}$, then $u \triangleright \exists \mathcal{A}_{q,q'}.\top$. In the case of 0 occurrences, either no witness at all is necessary or there is a witnessing walk without the new edges—in both cases, the same conditions hold in $\mathcal{W}$ already. Assume the IH for n. Consider $u \in (\exists \mathcal{A}_{q,q'}.\top)^{\mathcal{I}'}$, witnessed by the walk $u, r_1, u_1, \ldots, r_k, u_k$ that contains $n+1$ occurrences of edges from $\{e_1, e_2\}$. Let $\langle u_\ell, u_{\ell+1}\rangle$ be the first such occurrence. Let $(u_\ell, q_\ell, s, q_{\ell+1}, u_{\ell+1})$ be the transition of the run that corresponds to crossing this edge. The indices $q_\ell, q_{\ell+1}$ have been chosen for convenience and do not imply that this is the ℓ-th transition of the run. There is an accepting run of $\mathcal{A}_{q_{\ell+1},q'}$ on $u_{\ell+1}, r_{\ell+2} \ldots, r_k, u_k$ with n occurrences of edges from $\{e_1, e_2\}$. Thus by the IH, $u_{\ell+1} \triangleright \exists \mathcal{A}_{q_{\ell+1},q'}.\top$. We claim that $u_\ell \in (\exists \mathcal{A}_{q_\ell,q'}.\top)^{\mathcal{I}}$. Consider two cases.

Case 1. $\langle u_\ell, u_{\ell+1}\rangle = e_1$. We have $u_\ell = w$, $u_{\ell+1} = vi$. Because $vi \approx wi$, we have $wi \triangleright \exists \mathcal{A}_{q_{\ell+1},q'}.\top$. Since $\mathcal{I} \models \mathcal{T}$, thus $wi \in (\exists \mathcal{A}_{q_{\ell+1},q'}.\top)^{\mathcal{I}}$. Now from the fact that $\langle w, wi\rangle$ and $\langle w, vi\rangle$ are of the same role, it follows that $w \in (\exists \mathcal{A}_{q_\ell,q'}.\top)^{\mathcal{I}}$.

Case 2. $\langle u_\ell, u_{\ell+1}\rangle = e_2$. We have $u_\ell = vi0^{k-1}$, $u_{\ell+1} = w$. Because $w \approx v$, we have $wi \triangleright \exists \mathcal{A}_{q_{\ell+1},q'}.\top$. Since $\mathcal{I} \models \mathcal{T}$, it holds that $v \in (\exists \mathcal{A}_{q_{\ell+1},q'}.\top)^{\mathcal{I}}$. The edges $\langle vi0^{k-1}, w\rangle$ and $\langle vi0^{k-1}, w\rangle$ are of the same role, hence $vi0^{k-1} \in (\exists \mathcal{A}_{q_\ell,q'}.\top)^{\mathcal{I}}$.

Now, as our walk contains no occurrences of the new edges up to u_ℓ, the prefix $u, r_1, u_1, \ldots, r_\ell, u_\ell$ is present in $\mathcal{I}$. Therefore, $u \in (\exists \mathcal{A}_{q,q'}.\top)^{\mathcal{I}}$, and again by the fact that $\mathcal{I} \models \mathcal{T}$, we conclude that $u \triangleright \exists \mathcal{A}_{q,q'}.\top$.

Cycle-demand constraints. Observe that every self-loop on a vertex in $\mathcal{W}$ is also present in $\mathcal{W}'$. No self-loops are added thanks to the fact that $vi \neq w$. Thus, constraints with the right-hand side of the form $\exists r.\mathsf{Self}$ are satisfied. Now, for one direction, assume $u \triangleright \exists r^+.\mathsf{Self}$. If $\langle u, u\rangle \in S(r)$, we achieve our goal immediately. Otherwise, by integrity of $\mathcal{W}$, there is a witnessing cycle $u, r, u'j, r, u'j0, \ldots, r, u'j0^{k-1}, r, u$ in $\mathcal{I}$, where u is a descendant of $u'j$ if $u' \neq u$. Just like in the case of walk-demand constraints, if it contains no element of T_{wi}, then it is present in $\mathcal{I}'$ as well. Assume, therefore, that this cycle contains an element of T_{wi}. Then, $u'j \in T_{wi}$, because $i \neq 0$. Now, by integrity again, if $u \neq u'$, then u is a descendant of $u'j$. If $u \neq u'$ held, then u would belong to T_{wi}, a contradiction. Hence, $u = u'$. Now, because $uj \in T_{wi}$ and $u \in \Delta^{\mathcal{I}'}$, we obtain $uj = wi$. It suffices to show that $w \in (\exists s^+.\mathsf{Self})^{\mathcal{I}'}$. We choose the walk $w, s, vi, s, vi0, \ldots, s, vi0^{k-1}, s, w$, hence proven. □

B.2. Proof of Lemma 5.7

Proof. By the *extended type* of the i-th vertex on a path, we mean the tuple consisting of: the types $\lambda(v_i)$ and $\lambda(v_{i+1})$, the unique role name r_i such that $(v_i, v_{i+1}) \in r_i^{\mathcal{I}}$, the number $d \leqslant N$ such that v_{i+1} is the d-th child of v_i, and the collections of values of traces $\mathrm{tr}_r(v_i)$ and $\mathrm{tr}_r(v_{i+1})$ for each $r \in \mathbf{N}_{\mathbf{R}}^{\circlearrowleft}$.

By Lemma 5.3, there are only finitely many such extended types. What is more, all 0-paths are finite. Hence, by the pigeonhole principle, infinitely many vertices on the path share the same extended type and are non-zero children of their parent. To conclude the lemma, it suffices to select two such vertices at distance at least two. □

B.3. Proofs of the observations from Lemma 5.8

Proof.

1. $\varepsilon \in X$. *Proof:* Suppose otherwise. Then, there is a maximal n, say n', such that there is some vertex w at depth n in T_n. Consequently, $T_{n'+1}$ has no vertex at depth $n'+1$, contradicting our assumption.

2. If $v \in X$, then $vi \in X$ for some i. *Proof:* Suppose $v \in X$ and no child of v belongs to X. Then, for every i, let us define n_i to be the maximal value of n such that vi has in its subtree a vertex at depth n in T_n ($|v|$ if vi does not exist and $|vi|$ if vi has no children). Now since the tree has finite degree, we can choose some i such that n_i is maximal. The vertex v has no vertex in its subtree at depth n_i+1 in T_{n_i+1}, a contradiction.

3. If $vi \in X$, then the path from the root to v contains no edge similar to $\langle v, vi\rangle$ that does not directly precede it. *Proof:* This clearly follows from the definition of the wrapping process. If $\langle v, vi\rangle$ is similar to some edge that comes before it (but does not directly precede it) on the aforementioned path, then vi is removed at stage $|vi|$ of the wrapping process and its subtree does not exist in $T_{|vi|}$.

□