Decidability and Complexity of $\mathcal{ALCOIF}$ with Transitive Closure (and More)

Jean Christoph Jung[1] and Carsten Lutz[1] and Thomas Zeume[2]

[1] Universität Bremen, Germany, {jeanjung,clu}@uni-bremen.de
[2] TU Dortmund, Germany, thomas.zeume@cs.tu-dortmund.de

Abstract. We prove that satisfiability and finite satisfiability in the description logic $\mathcal{ALCOIF}_{reg}$ are NExpTime-complete when every regular role expression of the form α^* contains either no functional role or only a single role name (and possibly its inverse). Notably, this encompasses the extension of $\mathcal{ALCOIF}$ with transitive closure of roles and the modal logic of linear orders and successor, with converse.

1 Introduction

There has been a long quest for description logics (DLs) that are as expressive as possible while still being decidable. This has resulted in a prominent family of very expressive DLs such as $\mathcal{ALCOIF}$ and $\mathcal{SHOIQ}$ that support nominals, inverse roles, and functional roles or generalizations thereof. The combination of these three expressive means is technically challenging but also conceived as being very useful in applications. In fact, the member $\mathcal{SROIQ}$ of this family of DLs has been standardized as the OWL 2 DL ontology language by the W3C [15, 22].

A natural feature of DLs that has not been included in OWL 2 is a transitive closure operator on roles and, more generally, regular expressions over roles [1, 2, 5]. The decidability status of very expressive DLs with transitive closure is somewhat unclear. Decidability of the extension of $\mathcal{SHOIQ}$ with transitive closure has been claimed in [8], but according to personal communication with the authors there are problems in the construction and a corrected version has not yet been published. In this paper, we prove that satisfiability in the extension of $\mathcal{ALCOIF}$ with transitive closure is decidable where '$\mathcal{F}$' refers to global functionality restrictions, that is, roles can be declared to be partial functions in the TBox. We establish the same result for finite satisfiability (which does not coincide with the unrestricted case) and show that both problems are in NExpTime, thus NExpTime-complete and not harder than in the case without transitive closure. Our decision procedures are based on a decomposition of the TBox into parts that are only loosely related, and on reductions to satisfiability in $\mathcal{ALCOI}$ with regular role expressions and to the existential fragment of Presburger arithmetic.

We also make a step towards regular role expressions in $\mathcal{ALCOIF}$. In the results stated above, we in fact admit such expressions under the restriction that

every regular role expression of the form α^* contains either no functional role or only a single role name (and possibly its inverse). While this certainly is a strong restriction, the extension still provides non-trivial expressive power. For example, it makes it possible to speak about the length of role paths modulo a constant and to express the until operator of temporal logic. The decidability of unrestricted $\mathcal{ALCOIF}$ with regular role expressions remains an open problem. One may take it as an indication of being close to undecidability that satisfiability in $\mathcal{ALCOIF}$ extended with ω-regular expressions is undecidable [28].

Apart from the DL view, there are other interesting perspectives on our results. One is related to propositional dynamic logic (PDL). It is known that satisfiability is decidable and ExpTime-complete in the extension of PDL with any two of nominals, converse, and functional relations [6, 10, 11]. For the combination of all three, decidability is open. Our result can be viewed as showing decidability in a special case, that is, under the syntactic restriction given above. A related frontier of undecidability is that satisfiability in the μ-calculus is undecidable when nominals, converse, and functional relations are added [4].

Another interesting perspective is provided by the fact that $\mathcal{ALCOIF}$ is an expressive fragment of C^2 [12, 23, 24], two variable first-order logic with counting quantifiers, and in fact even of the guarded fragment GC^2 of C^2 [25]. It is known that C^2 easily becomes undecidable when (an unrestricted number of) relations can be declared to have special semantic properties such as being a linear order [21], a transitive relation [13], or an equivalence relation [18]; see also [19] for an overview. In fact, the same is true for GC^2. For example, GC^2 is undecidable when two equivalence relations are added [26] and also when three linear orders are added that can only be used in guards [17]. Decidability can sometimes be achieved when only a limited number of special relations is admitted. For example, finite satisfiability is decidable in C^2 extended with a single equivalence relation [26] and in the two variable fragment without counting when two linear orders are added [29, 34]. In the logic studied in this paper, it is possible to express that a role is transitive, an equivalence relation, a linear order, or a forest, respectively, possibly using fresh auxiliary symbols. Thus, our results show that (finite) satisfiability of $\mathcal{ALCOIF}$ remains decidable when we admit that an unbounded number of relations is declared to have any of the mentioned semantic properties, in marked contrast to C^2 and GC^2. We remark that finite satisfiability in $\mathcal{ALCOIQ}$ extended with forests has been shown in [20]. Our results capture the $\mathcal{ALCOIF}$ fragment of this DL, reprove decidability in NExpTime of finite satisfiability and establish the same upper bound for unrestricted satisfiability.

2 Preliminaries

Let $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$ be countably infinite sets of *concept, role, and individual names*. In $\mathcal{ALCOI}_{reg}$, *concepts* C and *(regular) roles* α are defined by

$$C ::= \top \mid A \mid \{a\} \mid \neg C \mid C \sqcap C \mid \exists \alpha.C \quad \alpha ::= C? \mid r \mid r^- \mid \alpha^* \mid \alpha \cdot \alpha \mid \alpha + \alpha$$

where A ranges over $\mathsf{N_C}$, a over $\mathsf{N_I}$, and r over $\mathsf{N_R}$. We use $C \sqcup D$ to abbreviate $\neg(\neg C \sqcap \neg D)$, refer to a concept of the form $\{a\}$ as a *nominal*, and to a role of the form $C?$ as a *test*. An $\mathcal{ALCOIF}_{reg}$-TBox is a finite set of *concept inclusions* $C \sqsubseteq D$ with C, D $\mathcal{ALCOI}_{reg}$-concepts and *functionality assertions* $\mathsf{func}(R)$ with R a role name or of the form r^- with r a role name. A *concept definition* takes the form $A \equiv C$ with A a concept name and C an $\mathcal{ALCOI}_{reg}$-concept, and it is an abbreviation for the concept inclusion $\top \sqsubseteq (\neg A \sqcup C) \sqcap (\neg C \sqcup A)$. We use $\mathsf{RN_f}(\mathcal{T})$ to denote the set of role names r such that $\mathsf{func}(r) \in \mathcal{T}$ or $\mathsf{func}(r^-) \in \mathcal{T}$. For the semantics, we refer to [2]. When speaking about *(finite) satisfiability*, we generally mean the satisfiability of a TBox.

Throughout the paper, we adopt a syntactic restriction on roles that is essential for our approach to work. An $\mathcal{ALCOIF}^-_{reg}$-TBox $\mathcal{T}$ is an $\mathcal{ALCOIF}_{reg}$-TBox that satisfies the following condition:

(∗) if the role α^* occurs in $\mathcal{T}$ and the role name $r \in \mathsf{RN_f}(\mathcal{T})$ occurs in α outside of tests, then no other role name occurs in α outside of tests.

Note that $\mathcal{ALCOIF}^-_{reg}$ contains the natural extension $\mathcal{ALCOIF}^+$ of $\mathcal{ALCOIF}$ with transitive closure of roles, without any syntactic restrictions on the latter. We give some examples that further illustrate the expressive power of $\mathcal{ALCOIF}^-_{reg}$.

Example 1. (1) $A \sqsubseteq \exists(r \cdot r \cdot r)^*.B$ expresses that every A can reach a B along some r-path of length divisible by 3. Adding $\mathsf{func}(r)$ does not violate (∗).

(2) $A \sqsubseteq \exists(C? \cdot r)^*.B$ expresses that every A can reach a B along an r-path on which C is true at every node, similar to the until operator of temporal logic. Adding $\mathsf{func}(r)$ does not violate (∗).

(3) $A \sqsubseteq \exists(r + r^-)^*.B$ expresses that whenever A is true, then B is true somewhere in the same maximal connected component of the role name r. Adding $\mathsf{func}(r)$ does not violate (∗)

(4) An $\mathcal{ALCOIF}^-_{reg}$-TBox $\mathcal{T}$ is finitely satisfiable iff $\mathcal{T} \cup \{\top \sqsubseteq \exists(r^-)^*.\{a\} \sqcap \exists r^*.\{b\},\ \mathsf{func}(r)\}$ is satisfiable, where the role name r and individual names a, b are fresh. Thus, finite satisfiability can be reduced to unrestricted satisfiability in polynomial time.

(5) We can simulate a transitive relation by the regular role $r \cdot r^*$ and an equivalence relation by the regular role $(r^- + r)^*$. We can enforce that the role name r is interpreted as a forest (of potentially infinite trees) by $\top \sqsubseteq \exists(r^-)^*.\forall r^-.\bot,\ \mathsf{func}(r^-)$.

(6) $\top \sqsubseteq \exists r^*.\{a\} \sqcup \exists(r^-)^*.\{a\},\ \{a\} \sqsubseteq \neg\exists r.\exists r^*.\{a\},\ \mathsf{func}(r),\ \mathsf{func}(r^-)$ enforces that $r \cdot r^*$ is a (strict) linear order with successor relation r.

In the remainder of the paper, it will be convenient to work with TBoxes $\mathcal{T}$ in *normal form*, meaning that $\mathcal{T}$ is a finite set of concept definitions of the form

$$A \equiv \{a\} \qquad A \equiv \neg B \qquad A \equiv B \sqcup B' \qquad A \equiv B \sqcap B' \qquad A \equiv \exists \alpha.B$$

where A, B, B' are concept names and of functionality assertions such that $\mathsf{func}(r^-) \in \mathcal{T}$ implies $\mathsf{func}(r) \in \mathcal{T}$ and every concept equation $A \equiv \exists \alpha.B \in \mathcal{T}$ satisfies the following conditions:

(i) if α contains a role name $r \in \mathsf{RN_f}(\mathcal{T})$, then it contains no other role name;
(ii) for every test $C?$ that occurs in α, C is a concept name;
(iii) if $\alpha = \beta + \beta'$, then $\mathcal{T}$ contains $A_1 \equiv \exists\beta.B$, $A_2 \equiv \exists\beta'.B$, $A \equiv A_1 \sqcup A_2$ for some A_1, A_2;
(iv) if $\alpha = \beta \cdot \beta'$, then $\mathcal{T}$ contains $A \equiv \exists\beta.A_1$, $A_1 \equiv \exists\beta'.B$, for some A_1;
(v) if $\alpha = \beta^*$, then $\mathcal{T}$ contains $A \equiv B \sqcup A_1$, $A_1 \equiv \exists\beta.A$, for some A_1.

Conditions (iii)-(v) are inspired by the Fischer-Ladner closure in PDL [9]. The following lemma shows that we can work with TBoxes in normal form without loss of generality.

Lemma 1. *Every $\mathcal{ALCOIF}^-_{reg}$-TBox $\mathcal{T}$ can be converted in polynomial time into a TBox $\mathcal{T}'$ in normal form such that $\mathcal{T}$ is (finitely) satisfiable iff $\mathcal{T}'$ is (finitely) satisfiable.*

Presburger Arithmetic. Presburger arithmetic is the first-order (FO) theory of the non-negative integers with addition $\mathrm{Th}(\mathbb{N}, 0, 1, +)$ [14, 27]. More precisely, a *term* is a constant 0 or 1, a variable, or the sum $t_1 + t_2$ of terms t_1, t_2. A *Presburger formula* is an FO formula over atoms of the form $t_1 = t_2$ with t_1, t_2 terms. An *existential* Presburger formula is a Presburger formula of the shape $\exists\mathbf{x}\,\varphi(\mathbf{x}, \mathbf{y})$ where φ is quantifier-free, and $\mathbf{x}, \mathbf{y}$ are tuples of variables. If $\varphi(\mathbf{y})$ is a Presburger formula with free variables $\mathbf{y}$ and $\mathbf{a} \in \mathbb{N}$ has the same arity as $\mathbf{y}$, then $\mathbf{a}$ is a *model of* φ if $\mathbb{N} \models \varphi(\mathbf{a})$. We use $\mathsf{Mod}(\varphi)$ to denote the set of all models of φ and say that φ is *satisfiable* if $\mathsf{Mod}(\varphi) \neq \emptyset$.

3 Decomposing TBoxes

As a foundation for our decision procedures, we show that (finite) satisfiability of an $\mathcal{ALCOIF}^-_{reg}$-TBox $\mathcal{T}$ can be decided by checking the (finite) satisfiability of certain subsets of $\mathcal{T}$ while ensuring a rather modest form of synchronization via the multiplicity of types. An important feature of this decomposition is that each of the subsets is either an $\mathcal{ALCOI}_{reg}$-TBox or an $\mathcal{ALCOIF}^-_{reg}$-TBox that contains only a single role name.

Let $\mathcal{T}$ be an $\mathcal{ALCOIF}^-_{reg}$-TBox in normal form. Let

- $\mathcal{T}_{bool}$ denote all concept definitions in $\mathcal{T}$ that are not of the form $A \equiv \exists\alpha.B$,
- $\mathcal{T}_{reg}$ denote the set of all concept definitions $A \equiv \exists\alpha.B$ in $\mathcal{T}$ such that no $r \in \mathsf{RN_f}(\mathcal{T})$ occurs in α, and
- $\mathcal{T}_r$ denote the set of all $A \equiv \exists\alpha.B$ such that $r \in \mathsf{RN_f}(\mathcal{T})$ is the only role name that occurs in α, plus $\mathcal{T} \cap \{\mathsf{func}(r), \mathsf{func}(r^-)\}$.

Then $\mathcal{T}_{bool} \cup \mathcal{T}_{reg}$ is an $\mathcal{ALCOI}_{reg}$-TBox and every TBox $\mathcal{T}_{bool} \cup \mathcal{T}_r$ contains no role name other than r.

A *type for* $\mathcal{T}$ is a set t of concept names used in $\mathcal{T}$. For an interpretation $\mathcal{I}$ and $d \in \Delta^{\mathcal{I}}$, the *type realized by* d is $\mathsf{tp}_{\mathcal{I}}(d) = \{A \text{ used in } \mathcal{T} \mid d \in A^{\mathcal{I}}\}$. We say that $\mathcal{I}$ *realizes* the set of types Γ if $\Gamma = \{\mathsf{tp}_{\mathcal{I}}(d) \mid d \in \Delta^{\mathcal{I}}\}$. For each type t, we denote with $\#_t(\mathcal{I})$ the cardinality of the set $\{d \in \Delta^{\mathcal{I}} \mid \mathsf{tp}_{\mathcal{I}}(d) = t\}$, writing $\#_t(\mathcal{I}) = \infty$ if t is realized infinitely often in $\mathcal{I}$.

Proposition 1. *An $\mathcal{ALCOIF}^{-}_{reg}$-TBox $\mathcal{T}$ is (finitely) satisfiable iff there is a set Γ of types for $\mathcal{T}$ such that*

1. *there is a model $\mathcal{I}_{reg}$ of $\mathcal{T}_{bool} \cup \mathcal{T}_{reg}$ that realizes Γ;*
2. *there are (finite) interpretations $\mathcal{I}_r$, $r \in \mathsf{RN_f}(\mathcal{T})$ such that, for all $r, s \in \mathsf{RN_f}(\mathcal{T})$:*
   (a) $\mathcal{I}_r \models \mathcal{T}_{bool} \cup \mathcal{T}_r$;
   (b) $\mathcal{I}_r$ realizes Γ;
   (c) $\#_t(\mathcal{I}_r) = \#_t(\mathcal{I}_s)$ for all $t \in \Gamma$.

4 Finite Satisfiability

The characterization provided in Proposition 1 gives rise to a transparent decision procedure for finite satisfiability via a reduction to satisfiability in $\mathcal{ALCOI}_{reg}$ (to check Condition 1) and to satisfiability in the existential fragment of Presburger arithmetic (to check the finite version of Condition 2). This reduction yields a NExpTime upper bound and since finite satisfiability in $\mathcal{ALCOIF}$ is NExpTime-hard [31], we obtain the following.

Theorem 1. *Finite satisfiability in $\mathcal{ALCOIF}^{-}_{reg}$ is* NExpTime*-complete.*

The central observation for checking Condition 2 via Presburger arithmetic is the following.

Lemma 2. *Let $\{t_1, \dots, t_n\}$ be the set of all types for $\mathcal{T}$. For every $r \in \mathsf{RN_f}(\mathcal{T})$, one can construct in time single exponential in $\mathcal{T}$ an existential Presburger formula $\varphi_{\mathcal{T},r}$ with n free variables such that, for all $\mathbf{a} = (a_1, \dots, a_n) \in \mathbb{N}^n$, the following are equivalent:*

1. *$\mathbf{a} \in \mathsf{Mod}(\varphi_{\mathcal{T},r})$;*
2. *there is a model $\mathcal{I}$ of $\mathcal{T}_{bool} \cup \mathcal{T}_r$ such that $\#_{t_i}(\mathcal{I}) = a_i$, for all $i \in \{1, \dots, n\}$.*

Before proving Lemma 2, let us first summarize how it yields the upper bound in Theorem 1. We guess a set Γ of types for $\mathcal{T}$ and verify Conditions 1 and 2 from Proposition 1. Condition 1 is equivalent to satisfiability of the $\mathcal{ALCOI}_{reg}$-TBox

$$\widehat{\mathcal{T}} = \mathcal{T}_{bool} \cup \mathcal{T}_{reg} \cup \{\top \sqsubseteq \bigsqcup_{t \in \Gamma} \bigsqcap t\} \cup \{\top \sqsubseteq \exists \widehat{r}. \bigsqcap t \mid t \in \Gamma\},$$

where $\widehat{r}$ is a fresh role name and $\bigsqcap t$ denotes the conjunction of all $A \in t$ and $\neg A$ for all $A \notin t$ that occur in $\mathcal{T}$. Satisfiability in $\mathcal{ALCOI}_{reg}$ is ExpTime-complete [3], but we have to be careful since $\widehat{\mathcal{T}}$ is of size (single) exponential in the size of $\mathcal{T}$. Fortunately, a slight modification of the algorithm in [3] achieves that the runtime is single exponential only in the number of concept names and concepts of the form $\exists r.C$ in the input TBox, and the number of such concepts in $\widehat{\mathcal{T}}$ is polynomial in the size of $\mathcal{T}$.

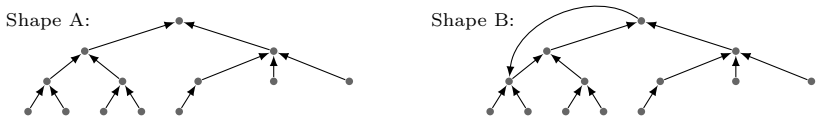

Fig. 1. Possible shapes of connected components of functional relations.

To verify Condition 2, for every $r \in \mathsf{RN}_\mathsf{f}(\mathcal{T})$ we construct the existential Presburger formula $\varphi_{\mathcal{T},r}(x_1, \ldots, x_n)$ from Lemma 2. Then, the sentence

$$\varphi = \exists \mathbf{x} \bigwedge_{r \in \mathsf{RN}_\mathsf{f}(\mathcal{T})} \varphi_{\mathcal{T},r}(x_1, \ldots, x_n) \wedge \bigwedge_{t_i \in \Gamma} (x_i > 0) \wedge \bigwedge_{t_i \notin \Gamma} (x_i = 0)$$

is satisfiable iff Condition 2 is true. It remains to note that satisfiability of existential Presburger formulas is NP-complete [33, 14].

The rest of the section is devoted to the proof of Lemma 2. A crucial observation is that if $\mathsf{func}(r) \in \mathcal{T}$, then according to the semantics $r^\mathcal{I}$ must have a rather restricted form in every model $\mathcal{I}$ of $\mathcal{T}$. More precisely, every maximal connected component of the directed graph $(\Delta^\mathcal{I}, r^\mathcal{I})$ takes one of the two forms depicted in Figure 1, that is, it is a tree after reversing the edges (Shape A), possibly with a single additional outgoing edge at the root (Shape B). This enables the construction of an automaton on finite trees whose language is the set of all finite models of $\mathcal{T}_{bool} \cup \mathcal{T}_r$, under a suitable encoding that (among other things) only implicitly represents the extra outgoing edge of the root in components of Shape B. Having these tree automata in place, we then exploit the close connection between tree automata and context free languages and the fact that the Parikh image of any context free language (which under our encoding describes type multiplicities) can be described by a Presburger formula [33].

Trees and Tree Automata. A *tree* is a prefix-closed subset $T \subseteq (\mathbb{N} \setminus \{0\})^*$. A node $w \in T$ is a successor of $v \in T$ and v is a predecessor of w if $w = v \cdot i$ for some $i \in \mathbb{N}$. A tree is *binary* if every inner node has exactly two successors. Let Σ be a finite alphabet. A *Σ-labeled tree* is a pair (T, τ) where T is a tree and $\tau : T \to \Sigma$ assigns a letter from Σ to each node in T. We will sometimes write only τ for the Σ-labeled tree (T, τ) if T is understood. By (T_n, τ_n), we denote the subtree of (T, τ) rooted at $n \in T$.

As our main automata model, we use *two-way alternating parity tree automata (2APTA)* over finite Σ-labeled binary trees [30, 32]. Note that there is no a priori bound on the degree of the structures in Figure 1. For the automata construction, however, we prefer to work with binary trees, relying on our encoding of interpretations as trees to bridge the gap. If both $\mathsf{func}(r) \in \mathcal{T}$ and $\mathsf{func}(r^-) \in \mathcal{T}$, then the components are actually paths or cycles rather than trees. To achieve a uniform construction, we also encode those as binary trees.

Formally, a 2APTA is a tuple $\mathfrak{A} = (Q, \Sigma, q_0, \delta, \rho)$ where Q is a set of *states*, $q_0 \in Q$ is the *initial state*, δ is a *transition function*, and $\rho : Q \to \mathbb{N}$ is a *priority*

function that assigns a priority to each state. The transition function δ maps each state q and input letter $\sigma \in \Sigma$ to a positive Boolean formula $\delta(q, \sigma)$ over the truth constants true and false and *transitions* of the form $p, \Diamond p, \Box p, \Diamond^- p$ with $p \in Q$. We use $L(\mathfrak{A})$ to denote the set of Σ-labeled trees that are accepted by $\mathfrak{A}$, see the appendix for details.

Automata Construction. Let $\mathsf{func}(r) \in \mathcal{T}$. We show how to construct a 2APTA that accepts precisely the models of $\mathcal{T}_{bool} \cup \mathcal{T}_r$, suitably encoded.

We first describe how interpretations that consist of disjoint components of Shape A or B can be represented as binary Σ-labeled trees, for a suitable Σ. Ideally, we would like to make the root of each component a successor of the root of the Σ-labeled tree and use a marker to represent the end point of the extra edge of components of Shape B. Since we work with binary trees, however, we have to introduce intermediate nodes labeled with a dummy symbol "$\circ$" to fit all components beneath the root and to fit all successors beneath any node inside a component. We use the alphabet

$$\Sigma = \{\circ\} \cup \{(t, M) \mid t \text{ a type for } \mathcal{T} \text{ and } M \in \{-, \mathsf{root}, \mathsf{loop}, \mathsf{src}, \mathsf{tgt}\}\}.$$

We use root to mark the root of components of Shape A, and loop and src to mark the root of components of Shape B, depending on whether the edge outgoing from the root also ends at the root or not. In the latter case, we use tgt to mark the target of the edge outgoing from the root.

A Σ-labeled tree (T, τ) is *well-formed* if (i) for every nominal $\{a\}$ in $\mathcal{T}$, there is a unique node n_a such that $\tau(n_a) = (t, M)$ and $A \in t$ for some $A \equiv \{a\} \in \mathcal{T}_{bool}$, (ii) there is some $n \in T$ with $\tau(n) = (t, M)$ for some M, and (iii) for all $n \in T$:

1. if $\tau(n) = (t, M)$ with $M \in \{\mathsf{root}, \mathsf{loop}\}$, then for all nodes n' below n, $\tau(n')$ has the form $\circ$ or $(t', -)$;
2. if $\tau(n) = (t, \mathsf{src})$, then for all nodes n' below n, $\tau(n')$ has the form $\circ$, $(t', -)$, or (t', tgt), and there is a unique node n' below n with $\tau(n')$ of the form (t', tgt);
3. if $\tau(n) = (t, M)$ with $M \in \{-, \mathsf{tgt}\}$, then there is a node n' above n with $\tau(n')$ of the form (t', M), $M \in \{\mathsf{root}, \mathsf{loop}, \mathsf{src}\}$.

A well-formed tree τ represents the following interpretation $\mathcal{I}_\tau$:

- $\Delta^{\mathcal{I}_\tau} = \{n \in T \mid \tau(n) \neq \circ\}$;
- $a^{\mathcal{I}_\tau} = n_a$ for all nominals $\{a\}$;
- $A^{\mathcal{I}_\tau} = \{n \in T \mid \tau(n) = (t, M) \text{ and } A \in t\}$ for all concept names A;
- $(n, n') \in r^{\mathcal{I}_\tau}$ if one of the following is satisfied:
  - $n \neq n'$, $n \in T_{n'}$, and all nodes on the path from n to n' are labeled $\circ$,
  - n is marked with src and $n' \in T_n$ is marked with tgt, or
  - $n = n'$ and n is marked with loop.

Conversely, whenever $\mathcal{I}$ is a finite model of $\mathsf{func}(r)$, then there is a finite, well-formed tree τ such that $\mathcal{I}$ and $\mathcal{I}_\tau$ are isomorphic. Note in particular that if some $d \in \Delta^{\mathcal{I}}$ has only a single r-predecessor e, then in τ the node that represents d can have one successor representing e and another dummy successor labeled "$\circ$". We next construct the desired 2APTAs.

Lemma 3. *For every $r \in \mathsf{RN_f}(\mathcal{T})$, one can construct in time polynomial in the size of $\mathcal{T}$ a 2APTA $\mathfrak{A}_r$ with polynomially many states such that*

$$L(\mathfrak{A}_r) = \{\tau \mid \tau \text{ is well-formed and } \mathcal{I}_\tau \models \mathcal{T}_{bool} \cup \mathcal{T}_r\}.$$

Proof. (sketch) The automaton $\mathfrak{A}_r$ is the intersection of a 2APTA $\mathfrak{A}_0$ that checks well-formedness of the input tree τ and a 2APTA $\mathfrak{A}_1$ that assumes well-formedness of τ and verifies that $\mathcal{I}_\tau \models \mathcal{T}_{bool} \cup \mathcal{T}_r$. The 2APTA $\mathfrak{A}_0$ is easy to construct, details are omitted.

The automaton $\mathfrak{A}_1$ sends a copy of itself to every node n of the input tree τ and verifies that when $\tau(n) = (t, X)$ and $A \equiv C \in \mathcal{T}_{bool} \cup \mathcal{T}_r$, then $A \in t$ iff $n \in C^{\mathcal{I}_\tau}$. For the definitions in $\mathcal{T}_{bool}$, this only requires a 'local' check of t. For concept definitions $A \equiv \exists\alpha.B \in \mathcal{T}_r$ the automaton needs to verify the (non-)existence of an α-path that starts at n and ends in a node whose type includes B. This is implemented by representing α as a finite automaton $\mathcal{B}$ on finite words and tracing runs of $\mathcal{B}$ through $\mathcal{I}_\tau$.

If $\mathsf{func}(r^-) \in \mathcal{T}_r$, then $\mathfrak{A}_1$ additionally ensures that every node has at least one successor labeled with "$\circ$". □

From 2APTAs to Presburger Arithmetic. It is well-known that every 2APTA $\mathfrak{A}$ can be translated to an equivalent *non-deterministic top-down automaton on finite trees (NTA)* $\mathfrak{B}$, incurring at most a single exponential blow-up in the number of states [30, 32]. Details on NTAs can be found in the appendix. Here, we only recall that the transition relation contains tuples of the form $(q, \sigma, q_1 \cdots q_m)$, meaning that when $\mathfrak{B}$ is in state q and reads symbol σ, then it can send the states $q_1, \ldots, q_m$ to the m successors of the current node. We can view $\mathfrak{B}$ as a *context free grammar (CFG)* G by interpreting the states as non-terminal symbols with the initial state being the start symbol, the input symbols as the terminal symbols, and each transition $(q, \sigma, q_1 \cdots q_m)$ as a grammar rule $q \to \sigma q_1 \ldots q_m$. Then for any tree τ accepted by $\mathfrak{B}$, there is a word accepted by G in which all symbols have the same multiplicities as in τ, and vice versa. We make use of the following result.

Theorem 2. *[33, Theorem 4] Given a CFG G over alphabet $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$, one can compute in linear time an existential Presburger formula $\varphi_G(x_1, \ldots, x_k)$ such that for all $\mathbf{a} \in \mathbb{N}^k$, we have $\mathbf{a} \in \mathsf{Mod}(\varphi)$ iff there is a word $w \in L(G)$ such that the number of occurrences of σ_i in w is a_i, for all $i \in \{1, \ldots, k\}$.*

Recall that $t_1, \ldots, t_n$ are all types for $\mathcal{T}$. For each $r \in \mathsf{RN_f}(\mathcal{T})$, let G_r be the CFG obtained from $\mathfrak{A}_r$ as described above, and let $\varphi_{G_r}(\mathbf{y})$ be the Presburger formula from Theorem 2, where $\mathbf{y}$ is the sequence of all variables y_σ with $\sigma \in \Sigma$ (assuming some fixed order). The formula $\varphi_{\mathcal{T},r}$ from Lemma 2 is then

$$\varphi_{\mathcal{T},r}(x_1, \ldots, x_n) := \exists \mathbf{y} \, \big(\varphi_{G_r}(\mathbf{y}) \wedge \bigwedge_{i=1}^{n} x_i = \sum_{M \in \{-,\mathsf{root},\mathsf{loop},\mathsf{src},\mathsf{tgt}\}} y_{(t_i,M)}\big).$$

5 Unrestricted Satisfiability

Unrestricted satisfiability can again be decided by Proposition 1, that is, by guessing a set of types Γ for the input TBox $\mathcal{T}$ and then verifying that Conditions 1 and 2 from that proposition are satisfied. For Condition 1, this amounts to the same $\mathcal{ALCOI}_{reg}$ satisfiability check as before. For Condition 2, the crucial difference is that the components in Figure 1 can now be infinite. A natural way to adapt the approach used in the previous section to check Condition 2 would thus be to replace 2APTA on finite trees with 2APTA on infinite trees, and to then translate the latter into existential sentences of the variant of Presburger arithmetic that also admits the value ω. It is, however, not clear how such a translation can be achieved. We thus pursue an alternative approach, based on the following slight refinement of Proposition 1.

Proposition 2. *An $\mathcal{ALCOIF}^-_{reg}$-TBox $\mathcal{T}$ is satisfiable iff there are disjoint sets $\Gamma_{\mathsf{fin}}, \Gamma_{\mathsf{inf}}$ of types for $\mathcal{T}$ such that*

1. *there is a model $\mathcal{I}_{reg}$ of $\mathcal{T}_{bool} \cup \mathcal{T}_{reg}$ that realizes $\Gamma_{\mathsf{fin}} \cup \Gamma_{\mathsf{inf}}$;*
2. *there are countable interpretations $\mathcal{I}_r$, $r \in \mathsf{RN_f}(\mathcal{T})$, s.t., for all $r, s \in \mathsf{RN_f}(\mathcal{T})$:*
   (a) *$\mathcal{I}_r \models \mathcal{T}_{bool} \cup \mathcal{T}_r$;*
   (b) *$\mathcal{I}_r$ realizes $\Gamma_{\mathsf{fin}} \cup \Gamma_{\mathsf{inf}}$;*
   (c) *$\#_t(\mathcal{I}_r) = \#_t(\mathcal{I}_s)$ for all $t \in \Gamma_{\mathsf{fin}}$;*
   (d) *$\#_t(\mathcal{I}_r) = \infty$ for all $t \in \Gamma_{\mathsf{inf}}$.*

The requirement of $\mathcal{I}_r$ being countable in Point 2 is harmless since $\mathcal{ALCOIF}^-_{reg}$ is a fragment of first-order logic with countably infinite conjunctions and disjunctions, which enjoys the downwards Löwenheim-Skolem property [16]. Initially, we thus guess two sets of types Γ_{fin} and Γ_{inf} instead of a single set Γ. The central technical observation to deal with Condition 2 is as follows.

Lemma 4. *Let $\Gamma_{\mathsf{fin}} = \{t_1, \dots, t_n\}$ and Γ_{inf} be disjoint sets of types for $\mathcal{T}$. For every $r \in \mathsf{RN_f}(\mathcal{T})$, there is a set $\mathcal{P}_{\mathcal{T},r}$ of existential Presburger formulas with n free variables s.t., for all $\mathbf{a} = (a_1, \dots, a_n) \in \mathbb{N}^n$, the following are equivalent:*

1. *$\mathbf{a} \in \mathsf{Mod}(\varphi)$ for some $\varphi \in \mathcal{P}_{\mathcal{T},r}$;*
2. *there is a model $\mathcal{I}$ of $\mathcal{T}_{bool} \cup \mathcal{T}_r$ that realizes exactly the types in $\Gamma_{\mathsf{fin}} \cup \Gamma_{\mathsf{inf}}$ such that $\#_{t_i}(\mathcal{I}) = a_i$ for $1 \leq i \leq n$ and $\#_t(\mathcal{I}) = \infty$ for all $t \in \Gamma_{\mathsf{inf}}$.*

Moreover, there is a non-deterministic exponential time procedure that, given $\mathcal{T}$, Γ_{fin}, Γ_{inf}, and r, generates exactly the formulas in $\mathcal{P}_{\mathcal{T},r}$ as possible outputs.

Before proving Lemma 4, we first observe that it yields the intended result.

Theorem 3. *Satisfiability in $\mathcal{ALCOIF}^-_{reg}$ is* NExpTime*-complete.*

Proof. The lower bound is inherited from $\mathcal{ALCOIF}$ [31]. For the upper bound, we guess disjoint sets $\Gamma_{\mathsf{fin}} = \{t_1, \dots, t_n\}, \Gamma_{\mathsf{inf}}$ and verify Conditions 1 and 2 from Proposition 2. Condition 1 can be treated as in the proof of Theorem 1. Observe that Condition 2 is satisfied iff there are formulas $\varphi_r \in \mathcal{P}_{\mathcal{T},r}$, for $r \in \mathsf{RN_f}(\mathcal{T})$

as in Lemma 4 such that the sentence $\varphi = \exists x_1 \dots \exists x_n \bigwedge_{r \in \mathsf{RN_f}(\mathcal{T})} \varphi_r(x_1, \dots, x_n)$ is satisfiable. It remains to note that these formulas can be 'guessed' using the non-deterministic procedure from Lemma 4, and that satisfiability of φ can be checked in non-deterministic exponential time. □

Now for the proof of Lemma 4. We start by encoding (potentially infinite) models $\mathcal{I}_r$ of $\mathcal{T}_{bool} \cup \mathcal{T}_r$ as infinite trees. The encoding is essentially the same as in the previous section except that now we also have to deal with models $\mathcal{I}_r$ that do not have a root, that is, in which some nodes have an infinite outgoing r-path. Informally, these are represented by choosing an arbitrary element d that has an infinite outgoing r-path, marking it with root, and 'folding' all nodes reachable from d via r below d—the folded nodes are marked with an additional marker fold. Formally, we use the extended alphabet $\Sigma' = \Sigma \cup \{(t, \mathsf{fold}) \mid t \text{ a type for } \mathcal{T}\}$ where Σ is as in the previous section. Under the new encoding, a Σ'-labeled tree is well-formed if it is well-formed in the sense of the previous section with the exception that nodes marked with root are allowed to have a single outgoing path marked with fold. The interpretation $\mathcal{I}_\tau$ associated to a well-formed Σ'-labeled tree τ is defined as in the previous section except that additionally $(n, n') \in r^{\mathcal{I}_\tau}$ if n is the predecessor of n' in T and n' is marked with fold. Conversely, whenever $\mathcal{I}$ is a countable model of $\mathsf{func}(r)$, then there is a well-formed tree τ such that $\mathcal{I}$ and $\mathcal{I}_\tau$ are isomorphic. One can construct a non-deterministic parity tree automaton (NPTA) $\mathcal{A}_r$ that accepts exactly the encodings of models of $\mathcal{T}_{bool} \cup \mathcal{T}_r$, by first constructing a 2APTA over infinite trees essentially as in the proof of Lemma 3 and then converting it into an NPTA, which is possible in exponential time [32].

Lemma 5. *For every $r \in \mathsf{RN_f}(\mathcal{T})$, one can construct in time single exponential in the size of $\mathcal{T}$ an NPTA $\mathfrak{A}_r$ over Σ' with exponentially many states such that $L(\mathfrak{A}_r) = \{\tau \mid \tau \text{ is well-formed and } \mathcal{I}_\tau \models \mathcal{T}_{bool} \cup \mathcal{T}_r\}$.*

Parity Tree Automata and Presburger Arithmetic. We aim to construct the Presburger formulas from Lemma 4 using the NPTAs from Lemma 5. The sets Γ_{fin} and Γ_{inf} give rise to disjoint subsets Σ_{fin} and Σ_{inf} of Σ', that is, $\Sigma_{\mathsf{fin}} = \{(t, M) \in \Sigma' \mid t \in \Sigma_{\mathsf{fin}}\}$ and likewise for Σ_{inf}.

Let $\mathfrak{A} = (Q, \Sigma, q_0, \delta, \rho)$ be an NPTA and let $\Sigma_{\mathsf{fin}}, \Sigma_{\mathsf{inf}}$ be disjoint subsets of Σ'. We denote with $\mathfrak{A}^q$ the variant of $\mathfrak{A}$ that has q as initial state. We are interested in the multiplicities of the symbols from Σ_{fin} and Σ_{inf} in trees accepted by $\mathfrak{A}$. In this context, it is convenient to think of the trees (T, τ) accepted by $\mathfrak{A}$ as being partitioned into several components. One component is the finite initial piece of T that is minimal with the property of containing all occurrences of symbols from Σ_{fin} and having only symbols from Σ_{inf} on the leaf nodes. Each leaf node is then the root of another component that takes the form of a potentially infinite tree and has only symbols from Σ_{inf}. Now, the initial piece can be described by an NTA on *finite trees* and thus translated into an existential Presburger formula as before while the multiplicity of all symbols in the other components is already known to be ∞ (in the overall tree) and thus we only need to ensure that these components exist.

An $\mathfrak{A}$-*obligation* is a triple $(q, \sigma, \Psi) \in Q \times \Sigma' \times 2^{\Sigma_{\mathsf{inf}}}$ such that there is a $\Sigma' \setminus \Sigma_{\mathsf{fin}}$-labeled tree $\tau \in L(\mathfrak{A}^q)$ with $\tau(\varepsilon) = \sigma$ and $\#_{\sigma'}(\tau) = \infty$ for all $\sigma' \in \Psi$. Informally, an $\mathfrak{A}$-obligation describes a component of a tree that is not the initial component, whose root is labeled with σ, and in which each symbol from Ψ occurs infinitely often while there is no restriction on the number of occurrences of other symbols from Σ_{inf}. Let $\mathcal{O}_{\mathfrak{A}}$ be the set of all $\mathfrak{A}$-obligations. An $\mathfrak{A}$-*obligation set* is a set $S = \{(q_1, \sigma_1, \Psi_1), \dots, (q_m, \sigma_m, \Psi_m)\} \subseteq \mathcal{O}_{\mathfrak{A}}$ such that $\Psi_1, \dots, \Psi_m$ is a partition of Σ_{inf} with possibly some Ψ_i being the empty set. Let $\mathcal{S}_{\mathfrak{A}}$ denote the set of all $\mathfrak{A}$-obligation sets. The number of obligations in each $\mathfrak{A}$-obligation set S is at most single exponential in the size of $\mathcal{T}$ and S can be represented in single exponential space. The number of $\mathfrak{A}$-obligation sets is at most double exponential in the size of $\mathcal{T}$.

Lemma 6. *Let $\mathfrak{A} = (Q, \Sigma', q_0, \delta, \rho)$ be an NPTA and let $\Sigma_{\mathsf{fin}} = \{\sigma_1, \dots, \sigma_k\}$ and Σ_{inf} be disjoint subsets of Σ'. Then there is a family $(\varphi_S)_{S \in \mathcal{S}_{\mathfrak{A}}}$ of formulas of existential Presburger arithmetic such that for every $\mathbf{a} = (a_1, \dots, a_k) \in \mathbb{N}^k$, the following are equivalent:*

1. *$\mathbf{a} \in \mathsf{Mod}(\varphi_S)$ for some $S \in \mathcal{S}_{\mathfrak{A}}$;*
2. *there is a $\tau \in L(\mathfrak{A})$ such that*
   (a) $\#_{\sigma_i}(\tau) = a_i$ for $1 \leq i \leq k$ and
   (b) $\#_{\sigma}(\tau) = \infty$ for every $\sigma \in \Sigma_{\mathsf{inf}}$.

Given $\mathfrak{A}$, Σ_{inf}, Σ_{fin}, and an $S \in \mathcal{S}_{\mathfrak{A}}$, φ_S can be constructed in polynomial time.

Given Lemma 6 it is not hard to provide the desired set of formulas $\mathcal{P}_{\mathcal{T},r}$ from Lemma 4. Moreover, the existence of the non-deterministic exponential time procedure that generates them is a consequence of (the last sentence in) Lemma 6, the fact that we can generate all candidates for $\mathfrak{A}$-obligation sets with a non-deterministic polynomial time procedure, and the fact proved in the appendix that given a triple $(q, \sigma, \Psi) \in Q \times \Sigma' \times 2^{\Sigma_{\mathsf{inf}}}$, it is decidable in NP whether (q, σ, Ψ) is an $\mathfrak{A}$-obligation.

6 Conclusions

The most interesting question left open in this paper is whether satisfiability in unrestricted $\mathcal{ALCOIF}_{reg}$ (equivalently: in PDL extended with nominals, inverse roles, and functional relations) is decidable. However, it even appears to be difficult to adapt the presented approach to more modest extensions of $\mathcal{ALCOIF}^{-}_{reg}$ such as local (unqualified) functionality restrictions. Another interesting extension is with role hierarchies, transitioning from $\mathcal{ALCOIF}_{reg}$ to $\mathcal{SHOIF}_{reg}$. It is known that adding role hierarchies over regular roles leads to undecidability [7] and it is not difficult to add to our approach role hierarchies restricted to role names and their inverses subject to the additional condition that functional roles do not have subroles. It is also interesting to note that adding guarded Boolean operators on roles, as typically indicated by the letter b in DL names, results in undecidability even when restricted to role names and their inverses [17].

Acknowledgements. Jung and Lutz were supported by ERC consolidator grant 647289 CODA.

References

1. Baader, F.: Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In: Proceedings of IJCAI. pp. 446–451. Morgan Kaufmann (1991)
2. Baader, F., Horrocks, I., Lutz, C., Sattler, U.: An Introduction to Description Logic. Cambridge University Press (2017)
3. Bonatti, P.A., Lutz, C., Murano, A., Vardi, M.Y.: The complexity of enriched μ-calculi. Logical Methods in Computer Science **4**(3) (2008)
4. Bonatti, P.A., Peron, A.: On the undecidability of logics with converse, nominals, recursion and counting. Artif. Intell. **158**(1), 75–96 (2004)
5. Calvanese, D., Eiter, T., Ortiz, M.: Regular path queries in expressive description logics with nominals. In: Proceedings of IJCAI. pp. 714–720 (2009)
6. De Giacomo, G., Lenzerini, M.: Boosting the correspondence between description logics and propositional dynamic logics. In: Proceedings of AAAI. pp. 205–212 (1994)
7. Duc, C.L., Lamolle, M.: Decidability of description logics with transitive closure of roles in concept and role inclusion axioms. In: Proceedings of DL. CEUR Workshop Proceedings, vol. 573. CEUR-WS.org (2010)
8. Duc, C.L., Lamolle, M., Curé, O.: A decision procedure for $\mathcal{SHOIQ}$ with transitive closure of roles. In: Proceedings of ISWC. LNCS, vol. 8218, pp. 264–279. Springer (2013)
9. Fischer, M.J., Ladner, R.E.: Propositional dynamic logic of regular programs. J. Comput. Syst. Sci. **18**(2), 194–211 (1979)
10. Giacomo, G.D.: Decidability of class-based knowledge representation formalisms. Ph.D. thesis, Universita di Roma "La Sapienza" (1995)
11. Giacomo, G.D., Lenzerini, M.: TBox and ABox reasoning in expressive description logics. In: Proceedings of KR. pp. 316–327. Morgan Kaufmann (1996)
12. Grädel, E., Otto, M., Rosen, E.: Two-variable logic with counting is decidable. In: Proceedings of LICS. pp. 306–317. IEEE Computer Society (1997)
13. Grädel, E., Otto, M., Rosen, E.: Undecidability results on two-variable logics. Arch. Math. Log. **38**(4-5), 313–354 (1999)
14. Haase, C.: A survival guide to presburger arithmetic. SIGLOG News **5**(3), 67–82 (2018)
15. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible SROIQ. In: Proceedings of KR. pp. 57–67 (2006)
16. Keisler, H.J.: Model Theory for Infinitary Logic: Logic with Countable Conjunctions and Finite Quantifiers. North Holland Publishing Company (1971)
17. Kieronski, E.: Decidability issues for two-variable logics with several linear orders. In: Proceedings of CSL. pp. 337–351 (2011)
18. Kieronski, E., Otto, M.: Small substructures and decidability issues for first-order logic with two variables. J. Symb. Log. **77**(3), 729–765 (2012)
19. Kieronski, E., Pratt-Hartmann, I., Tendera, L.: Two-variable logics with counting and semantic constraints. SIGLOG News **5**(3), 22–43 (2018), https://dl.acm.org/citation.cfm?id=3242958
20. Kotek, T., Simkus, M., Veith, H., Zuleger, F.: Extending ALCQIO with trees. In: Proceedings of LICS. pp. 511–522 (2015)
21. Otto, M.: Two variable first-order logic over ordered domains. J. Symb. Log. **66**(2), 685–702 (2001)

22. OWL Working Group, W.: OWL 2 Web Ontology Language: Document Overview. W3C Recommendation (27 October 2009), available at http://www.w3.org/TR/owl2-overview/
23. Pacholski, L., Szwast, W., Tendera, L.: Complexity results for first-order two-variable logic with counting. SIAM J. Comput. **29**(4), 1083–1117 (2000)
24. Pratt-Hartmann, I.: Complexity of the two-variable fragment with counting quantifiers. Journal of Logic, Language and Information **14**(3), 369–395 (2005)
25. Pratt-Hartmann, I.: Complexity of the guarded two-variable fragment with counting quantifiers. J. Log. Comput. **17**(1), 133–155 (2007)
26. Pratt-Hartmann, I.: The two-variable fragment with counting and equivalence. Mathematical Logic Quarterly **61**(6), 474–515 (2015)
27. Presburger, M.: Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. pp. 92—-101 (1929)
28. Rudolph, S.: Undecidability results for database-inspired reasoning problems in very expressive description logics. In: Proceedings of KR. pp. 247–257. AAAI Press (2016)
29. Schwentick, T., Zeume, T.: Two-variable logic with two order relations. Logical Methods in Computer Science **8**(1) (2012)
30. Slutzki, G.: Alternating tree automata. Theor. Comput. Sci. **41**, 305–318 (1985)
31. Tobies, S.: The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. J. Artif. Intell. Res. **12**, 199–217 (2000)
32. Vardi, M.Y.: Reasoning about the past with two-way automata. In: Proceedings of ICALP. pp. 628–641 (1998)
33. Verma, K.N., Seidl, H., Schwentick, T.: On the complexity of equational horn clauses. In: Proceedings of CADE. pp. 337–352 (2005)
34. Zeume, T., Harwath, F.: Order-invariance of two-variable logic is decidable. In: Proceedings of LICS. pp. 807–816 (2016)

Appendix

Proofs for Section 2

Lemma 1. *Every $\mathcal{ALCOIF}^{-}_{reg}$-TBox $\mathcal{T}$ can be converted in polynomial time into a TBox $\mathcal{T}'$ in normal form such that $\mathcal{T}$ is (finitely) satisfiable iff $\mathcal{T}'$ is (finitely) satisfiable.*

Proof. Let $\mathcal{T}$ be an $\mathcal{ALCOIF}^{-}_{reg}$-TBox. It is easy to achieve that $\mathsf{func}(r^-) \in \mathcal{T}$ implies $\mathsf{func}(r) \in \mathcal{T}$. In fact, if $\mathsf{func}(r^-) \in \mathcal{T}$ and $\mathsf{func}(r) \notin \mathcal{T}$, then we can swap r and r^- throughout $\mathcal{T}$, which clearly preserves (un)satisfiability.

We proceed in three steps. In the first step, we exhaustively apply the following rewrite steps:

- replace every concept $\exists\beta \cdot \beta'.C$ in $\mathcal{T}$ with $\exists\beta.\exists\beta'.C$ and
- replace every concept $\exists\beta + \beta'.C$ in $\mathcal{T}$ with $\exists\beta.A \sqcup \exists\beta'.A$ and add $A \equiv C$, with A a fresh concept name.

Call the resulting TBox $\mathcal{T}_1$. In the second step, we introduce a fresh concept name X_C for every concept C in $\mathcal{T}_1$ that is not a concept name. If D is a concept name, we use X_D to denote D. Put

$$\begin{aligned} \delta(\top) &= \top & \delta(A) &= A \\ \delta(\{a\}) &= X_{\{a\}} & \delta(\neg C) &= \neg X_C \\ \delta(C \sqcap D) &= X_C \sqcap X_D & \delta(\exists\alpha.C) &= \exists\beta.X_C \end{aligned}$$

where β is the result of replacing every test $D?$ in α that is not nested within another test with $X_D?$. Now $\mathcal{T}_2$ consists of the following:

- $X_{\{a\}} \equiv \{a\}$, for all $\{a\}$ that occur in $\mathcal{T}$;
- $X_C \equiv X_D \sqcap A$ for every $C \sqsubseteq D \in \mathcal{T}_1$, with A a fresh concept name;
- $X_C \equiv \delta(C)$ for every concept C that occurs in $\mathcal{T}_1$.

It can be verified that, at this Point, Conditions (i) and (ii) of the normal form are satisfied. In particular, when a concept $\exists\alpha.A$ occurs in $\mathcal{T}_2$, then α contains no role name from $\mathsf{RN}_\mathsf{f}(\mathcal{T}_2)$ or only a single role name due to the restriction $(*)$ and the first conversion step.

The third step ensures Conditions (iii) to (v) by exhaustively applying the following transformations:

1. if $A \equiv \exists(\beta + \beta').B$ is a concept definition in $\mathcal{T}_2$, then add $A_1 \equiv \exists\beta.B$, $A_2 \equiv \exists\beta'.B$, and $A \equiv A_1 \sqcup A_2$, for fresh concept names A_1, A_2;
2. if $A \equiv \exists(\beta \cdot \beta').B$ is a concept definition in $\mathcal{T}_2$, then add $A \equiv \exists\beta.A'$, $A' \equiv \exists\beta'.B$, for a fresh concept name A';
3. if $A \equiv \exists\beta^*.B$ is a concept definition in $\mathcal{T}_0$, then add $A \equiv B \sqcup A'$ and $A' \equiv \exists\beta.A$, for a fresh concept name A'.

The conversion clearly needs only polynomial time. The resulting TBox $\mathcal{T}'$ is in normal form and $\mathcal{T}$ is (finitely) satisfiable iff $\mathcal{T}'$ is (finitely) satisfiable, as required. □

Proofs for Section 3

Proposition 1. *An $\mathcal{ALCOIF}^{-}_{reg}$-TBox $\mathcal{T}$ is (finitely) satisfiable iff there is a set Γ of types for $\mathcal{T}$ such that*

1. *there is a model $\mathcal{I}_{reg}$ of $\mathcal{T}_{bool} \cup \mathcal{T}_{reg}$ that realizes Γ;*
2. *there are (finite) interpretations $\mathcal{I}_r$, $r \in \mathsf{RN_f}(\mathcal{T})$ such that, for all $r, s \in \mathsf{RN_f}(\mathcal{T})$:*
   (a) $\mathcal{I}_r \models \mathcal{T}_{bool} \cup \mathcal{T}_r$;
   (b) $\mathcal{I}_r$ realizes Γ;
   (c) $\#_t(\mathcal{I}_r) = \#_t(\mathcal{I}_s)$ for all $t \in \Gamma$.

Proof. ($\Rightarrow$) Let $\mathcal{I}$ be a (finite) model of $\mathcal{T}$, and let Γ be the set of types realized in $\mathcal{I}$. Then, $\mathcal{I}_{reg} = \mathcal{I}$ witnesses Condition 1 and $\mathcal{I}_r = \mathcal{I}$, $r \in \mathsf{RN_f}(\mathcal{T})$, witness Condition 2.

($\Leftarrow$) Let Γ be a set of types for $\mathcal{T}$ that satisfies Conditions 1 and 2, that is, there is a model $\mathcal{I}_{reg}$ of $\mathcal{T}_{reg}$ that realizes Γ and a family of countable (finite) interpretations $\mathcal{I}_r$, $r \in \mathsf{RN_f}(\mathcal{T})$ that satisfies Conditions 2(a)–(c). It is easy to see that $\mathcal{ALCOIF}_{reg}$ is a fragment of $\mathcal{L}_{\omega_1\omega}$, the extension of first-order logic with countably infinite conjunctions and disjunctions. Since $\mathcal{L}_{\omega_1\omega}$ has the downward Löwenheim-Skolem property [16], we can assume that $\mathcal{I}_{reg}$ and all $\mathcal{I}_r$ are countable (and still realize Γ).

We construct a model $\mathcal{I}$ of $\mathcal{T}$ as follows. If there is no functionality assertion in $\mathcal{T}$, take $\mathcal{I} = \mathcal{I}_{reg}$ and we are done. Otherwise, fix some $r \in \mathsf{RN_f}(\mathcal{T})$. By Condition 2(c) and since the interpretations $\mathcal{I}_r$, $r \in \mathsf{RN_f}(\mathcal{T})$, are all countable or all finite, for every $s \in \mathsf{RN_f}(\mathcal{T})$ there is a bijection $\pi_s : \Delta^{\mathcal{I}_s} \to \Delta^{\mathcal{I}_r}$ such that $\mathsf{tp}_{\mathcal{I}_r}(d) = \mathsf{tp}_{\mathcal{I}_s}(\pi_s(d))$ for all $d \in \Delta^{\mathcal{I}_s}$. For types $t, t' \in \Gamma$ and a role name s, we write $t \rightsquigarrow^{\mathcal{T}}_s t'$ if

1. $B \in t'$ and $A \equiv \exists s.B \in \mathcal{T}$ implies $A \in t$ and
2. $B \in t$ and $A \equiv \exists s^{-}.B \in \mathcal{T}$ implies $A \in t'$.

The interpretation $\mathcal{I}$ is now defined as follows:

$$
\begin{array}{lll}
\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}_r} & & \\
A^{\mathcal{I}} = A^{\mathcal{I}_r}, & & \text{for all } A \in \mathsf{N_C} \\
r^{\mathcal{I}} = r^{\mathcal{I}_r} & & \\
s^{\mathcal{I}} = \{(\pi_s(d), \pi_s(e)) \mid (d,e) \in s^{\mathcal{I}_s}\} & & \text{for all } s \in \mathsf{RN_f}(\mathcal{T}), s \neq r, \\
s^{\mathcal{I}} = \{(d,e) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \mathsf{tp}_{\mathcal{I}}(d) \rightsquigarrow^{\mathcal{T}}_s \mathsf{tp}_{\mathcal{I}}(e)\} & & \text{for all } s \notin \mathsf{RN_f}(\mathcal{T}).
\end{array}
$$

To show that $\mathcal{I} \models \mathcal{T}$, it suffices to show that $\mathcal{I} \models \mathcal{T}_{bool}$, $\mathcal{I} \models \mathcal{T}_{reg}$, and $\mathcal{I} \models \mathcal{T}_s$ for all $s \in \mathsf{RN_f}(\mathcal{T})$. By construction of $\mathcal{I}$ and Condition 2(a), $\mathcal{I} \models \mathcal{T}_{bool} \cup \mathcal{T}_r$. Moreover, $\mathcal{I} \models \mathcal{T}_s$ for every $s \in \mathsf{RN_f}(\mathcal{T})$ with $s \neq r$ since $\mathcal{I}_s$ and $\mathcal{I}$ restricted to $s^{\mathcal{I}}$ are isomorphic (witnessed by π_s) and $\mathcal{I}_s \models \mathcal{T}_s$. It remains to show that $\mathcal{I} \models \mathcal{T}_{reg}$. Let $A \equiv \exists\alpha.B \in \mathcal{T}_{reg}$. We show the two directions of the equivalence $A^{\mathcal{I}} = (\exists\alpha.B)^{\mathcal{I}}$ separately.

For "$\subseteq$", one first proves the following claim by induction on the structure of α.

Claim 1. For all roles α in $\mathcal{T}_{reg}$, all $d, e \in \Delta^{\mathcal{I}}$, and all $d', e' \in \Delta^{\mathcal{I}_{reg}}$ such that $\mathsf{tp}_{\mathcal{I}}(d) = \mathsf{tp}_{\mathcal{I}_{reg}}(d')$ and $\mathsf{tp}_{\mathcal{I}}(e) = \mathsf{tp}_{\mathcal{I}_{reg}}(e')$, $(d', e') \in \alpha^{\mathcal{I}_{reg}}$ implies $(d, e) \in \alpha^{\mathcal{I}}$.

Proof of Claim 1. The induction base follows from the fact that $\mathsf{tp}_{\mathcal{I}_{reg}}(d) \rightsquigarrow_s \mathsf{tp}_{\mathcal{I}_{reg}}(e)$ for all $(d, e) \in s^{\mathcal{I}_{reg}}$. The induction step is then immediate.

Now let $d \in A^{\mathcal{I}}$. Since $\mathcal{I}_{reg}$ and $\mathcal{I}_r$ both realize Γ, there is an element $d' \in \Delta^{\mathcal{I}_{reg}}$ such that $\mathsf{tp}_{\mathcal{I}}(d) = \mathsf{tp}_{\mathcal{I}_{reg}}(d')$. Since $\mathcal{I}_{reg} \models \mathcal{T}_{reg}$, there is some e' such that $(d', e') \in \alpha^{\mathcal{I}_{reg}}$ and $e' \in B^{\mathcal{I}_{reg}}$. There is an element e with $\mathsf{tp}_{\mathcal{I}}(e) = \mathsf{tp}_{\mathcal{I}_{reg}}(e')$. Claim 1 yields $d \in (\exists \alpha.B)^{\mathcal{I}}$.

For the "$\supseteq$" direction, we prove by induction on α that for all $A \equiv \exists \alpha.B \in \mathcal{T}_{reg}$, $d \in (\exists \alpha.B)^{\mathcal{I}}$ implies $d \in A^{\mathcal{I}}$:

- If α is a test $X?$, then X is a concept name X, $d \in X^{\mathcal{I}}$, and $d \in B^{\mathcal{I}}$. Since $\mathcal{I}_{reg}$ and $\mathcal{I}_r$ both realize Γ, there is some $d' \in \Delta^{\mathcal{I}_{reg}}$ with $\mathsf{tp}_{\mathcal{I}}(d) = \mathsf{tp}_{\mathcal{I}_{reg}}(d')$. From $\mathcal{I}_{reg} \models \mathcal{T}_{reg}$, we obtain $d' \in A^{\mathcal{I}_{reg}}$ and thus $d \in A^{\mathcal{I}}$.
- If $\alpha = s$ or $\alpha = s^-$, for some role name s, then the statement follows from the definition of $\rightsquigarrow_s^{\mathcal{T}}$.
- If $\alpha = \beta \cdot \beta'$, then there is some $e \in \Delta^{\mathcal{I}}$ such that $(d, e) \in \beta^{\mathcal{I}}$ and $e \in (\exists \beta'.B)^{\mathcal{I}}$. By the normal form, $\mathcal{T}_{reg}$ contains concept definitions $A \equiv \exists \beta.A_1$ and $A_1 \equiv \exists \beta'.B$. By induction, we have $e \in A_1^{\mathcal{I}}$, and, again by induction, also $d \in A^{\mathcal{I}}$.
- If $\alpha = \beta + \beta'$, then $d \in (\exists \beta.B)^{\mathcal{I}}$ or $d \in (\exists \beta'.B)^{\mathcal{I}}$. By the normal form, $\mathcal{T}_{reg}$ contains concept definitions $A_1 \equiv \exists \beta.B$, $A_2 \equiv \exists \beta'.B$, and $A \equiv A_1 \sqcup A_2$. By induction, we have $d \in A_1^{\mathcal{I}}$ or $d \in A_2^{\mathcal{I}}$, and thus $d \in A^{\mathcal{I}}$.
- If $\alpha = \beta^*$, then $d \in (\exists \beta^i.B)^{\mathcal{I}}$, for some $i \geq 0$. Moreover, by the normal form, $\mathcal{T}_{reg}$ contains concept definitions $A_1 \equiv \exists \beta.A$ and $A \equiv B \sqcup A_1$. We show, by induction on i, that $d \in A^{\mathcal{I}}$. If $i = 0$, then $d \in B^{\mathcal{I}}$ and the statement follows from $A \equiv B \sqcup A_1 \in \mathcal{T}_{reg}$. If $i > 0$, there is an e with $(d, e) \in \beta^{\mathcal{I}}$ and $e \in (\exists \beta^{i-1}.B)^{\mathcal{I}}$. By induction, we have $e \in A^{\mathcal{I}}$. By the normal form, $d \in A_1^{\mathcal{I}}$ and also $d \in A^{\mathcal{I}}$.

Preliminaries for Tree Automata

Semantics of 2ATAs. A *run* of $\mathfrak{A}$ on a binary Σ-labeled tree (T, τ) is a $T \times Q$-labelled tree (T_r, r) such that $r(\varepsilon) = (\varepsilon, q_0)$ and whenever $x \in T_r$, $r(x) = (n, q)$, and $\delta(q, \tau(n)) = \theta$, then there is a subset S of the transitions that occur in θ such that S satisfies θ and:

- for every $p \in S$, there is a successor x' of x in T_r with $r(x') = (n, p)$;
- for every $\Diamond p \in S$, there is an $i \in \{1, 2\}$ and a successor x' of x in T_r with $r(x') = (n \cdot i, p)$;
- for every $\Box p \in S$ and all successors $n \cdot i \in T$ of n, there is a successor x' of x in T_r with $r(x') = (n \cdot i, p)$;

– for every $\Diamond^- p \in S$, there is a successor x' of x in T_r with $r(x') = (n \cdot (-1), p)$.

Let $\gamma = i_0 i_1 \cdots$ be an infinite path in T_r and denote, for $j \geq 0$, with q_j the state such that $r(i_j) = (x, q_j)$. The run γ is *accepting* if the largest number m such that $\rho(q_j) = m$, for infinitely many j, is even. A run T_r is accepting if every infinite path in T_r is accepting. The automaton accepts an input tree if there is an accepting run for it. We use $L(\mathfrak{A})$ to denote the set of trees accepted by $\mathfrak{A}$.

Non-deterministic tree automata (NTA). A NTA over finite Σ-labeled trees is a tuple $\mathfrak{A} = (Q, \Sigma, q_0, \Delta)$, where Q is a set of *states*, $q_0 \in Q$ is the *initial state*, and $\Delta \subseteq Q \times \Sigma \times Q^*$ is the *transition relation*. The semantics is defined in terms of runs. A *run* r on a Σ-labeled tree (T, τ) is a function $r : T \to Q$ such that

– $r(\varepsilon) = q_0$, and
– for every inner node $x \in T$ with successors $x_1, \ldots, x_n$ (in this order), there is a transition $(r(x), \tau(x), r(x_1) \cdots r(x_n)) \in \Delta$.

The language $L(\mathfrak{A})$ is the set of all finite trees which admit a run.

Non-deterministic Parity Tree Automata (NPTA). A NPTA over Σ-labeled trees is the extension of an NTA (Q, Σ, q_0, Δ) with a *priority function* $\rho : Q \to \mathbb{N}$. Runs are defined as for NTAs, and a run is *accepting* if every infinite path satisfies the parity condition specified by ρ, see the semantics for 2ATAs above. We denote with $L(\mathfrak{A})$ the set of all Σ-labeled trees τ that admit an accepting run.

Proofs for Section 4

We provide details for the 2ATA $\mathfrak{A}_1$ missing in the proof of Lemma 3.

A *finite automaton* is a tuple $\mathfrak{B} = (S, \Omega, s_0, \Delta, F)$, where S is a finite set of states, $s_0 \in S$ is the initial state, $F \subseteq S$ is a set of accepting states, $\Delta \subseteq S \times \Omega \times S$ is the transition relation, and Ω is the alphabet $\{r, r^-\} \cup \{A? \mid A? \text{ occurs in } \mathcal{T}\}$. For any $s \in S$, we use $\mathfrak{B}^s$ to denote the automaton obtained from $\mathfrak{B}$ by replacing the initial state with s.

For any interpretation $\mathcal{I}$, $\mathfrak{B}$ defines a binary relation $E^{\mathcal{I}}_{\mathfrak{B}}$ on $\Delta^{\mathcal{I}}$ defined by having $(d, e) \in E^{\mathcal{I}}_{\mathfrak{B}}$ iff there is a word $w = a_1 \ldots a_n \in L(\mathfrak{B})$ and a sequence $d_0, \ldots, d_n \in \Delta^{\mathcal{I}}$ with $d_0 = d$, $d_n = e$, and such that for every i with $0 \leq i \leq n$:

(i) if $a_i = A?$, then $d_{i-1} \in A^{\mathcal{I}}$ and $d_i = d_{i-1}$,
(ii) if $a_i = r$, then $(d_{i-1}, d_i) \in r^{\mathcal{I}}$, and
(iii) if $a_i = r^-$, then $(d_{i-1}, d_i) \in (r^-)^{\mathcal{I}}$.

It is well-known that for every regular role α, there is a finite automaton $\mathfrak{B}_\alpha$ of size polynomial in the size of α such that $\alpha^{\mathcal{I}} = E^{\mathcal{I}}_{\mathfrak{B}_\alpha}$, for every interpretation $\mathcal{I}$.

Lemma 3. *For every $r \in \mathsf{RN_f}(\mathcal{T})$, one can construct in time polynomial in the size of $\mathcal{T}$ a 2APTA $\mathfrak{A}_r$ with polynomially many states such that*

$$L(\mathfrak{A}_r) = \{\tau \mid \tau \text{ is well-formed and } \mathcal{I}_\tau \models \mathcal{T}_{bool} \cup \mathcal{T}_r\}.$$

Proof. We define $\mathfrak{A}_1 = (Q, \Sigma, q_0, \delta, \rho)$. The set Q consists of q_0, q_1, and states $\langle \mathfrak{B}^s_\alpha, B\rangle$, $\langle \mathfrak{B}^s_\alpha, \overline{B}\rangle$, $\langle \mathfrak{B}^s_\alpha, B, X\rangle$, $\langle \mathfrak{B}^s_\alpha, \overline{B}, X\rangle$, for every concept $\exists\alpha.B$ that occurs in $\mathcal{T}_r$, every state s in $\mathfrak{B}_\alpha$, and $X \in \Omega \cup \{L, R, \uparrow, \downarrow\}$.

The automaton visits every node in the input tree and verifies that it satisfies the concept definitions in $\mathcal{T}_{bool} \cup \mathcal{T}_r$. This process is initiated using the following transitions, for every $(t, M) \in \Sigma$. We write $t \models \mathcal{T}_{bool}$ if t satisfies all concept definitions in $\mathcal{T}_{bool}$.

$$
\begin{aligned}
\delta(q_0, \sigma) &= q_1 \wedge \Box q_0 && \text{if } \sigma \neq \circ\\
\delta(q_0, \circ) &= \Box q_0 &&\\
\delta(q_1, (t, M)) &= \mathsf{false} && \text{if } t \not\models \mathcal{T}_{bool}\\
\delta(q_1, (t, M)) &= \bigwedge_{A\in t, A\equiv\exists\alpha.B\in\mathcal{T}_r} \langle \mathfrak{B}_\alpha, B\rangle \wedge \bigwedge_{A\notin t, A\equiv\exists\alpha.B\in\mathcal{T}_r} \langle \mathfrak{B}_\alpha, \overline{B}\rangle && \text{if } t \models \mathcal{T}_{bool}
\end{aligned}
$$

If the automaton visits a node n in a state $\langle \mathfrak{B}, B\rangle$, it has the obligation to find a node $n' \in B^{\mathcal{I}_\tau}$ with $(n, n') \in E^{\mathcal{I}_\tau}_{\mathfrak{B}}$. Analogously, the visit of n in state $\langle \mathfrak{B}, \overline{B}\rangle$ is an obligation to verify that all nodes n' with $(n, n') \in E^{\mathcal{I}_\tau}_{\mathfrak{B}}$ do not satisfy $n' \in B^{\mathcal{I}_\tau}$. We include the following transitions for states $\langle \mathfrak{B}, B\rangle$ with $\mathfrak{B} = (S, \Omega, s_0, \Delta, F)$:

$$
\begin{aligned}
\delta(\langle \mathfrak{B}, B\rangle, (t, M)) &= \mathsf{true} && \text{if } s_0 \in F \text{ and } B \in t\\
\delta(\langle \mathfrak{B}, B\rangle, (t, M)) &= \textstyle\bigvee_{(s_0,a,s)\in\Delta} \langle \mathfrak{B}^s, B, a\rangle && \text{if } s_0 \notin F \text{ or } B \notin t
\end{aligned}
$$

In a state $\langle \mathfrak{B}, B, a\rangle$, the automaton has the obligation to simulate the step a in the represented interpretation and change to state $\langle \mathfrak{B}, B\rangle$. This is realized by the following transitions:

$$
\begin{aligned}
\delta(\langle \mathfrak{B}, B, A?\rangle, (t, M)) &= \langle \mathfrak{B}, B\rangle && \text{if } A \in t\\
\delta(\langle \mathfrak{B}, B, A?\rangle, (t, M)) &= \mathsf{false} && \text{if } A \notin t
\end{aligned}
$$

$$
\begin{aligned}
\delta(\langle \mathfrak{B}, B, r\rangle, (t, \mathsf{root})) &= \mathsf{false}\\
\delta(\langle \mathfrak{B}, B, r\rangle, (t, \mathsf{loop})) &= \langle \mathfrak{B}, B\rangle\\
\delta(\langle \mathfrak{B}, B, r\rangle, (t, \mathsf{src})) &= \Diamond\langle \mathfrak{B}, B, L\rangle\\
\delta(\langle \mathfrak{B}, B, r\rangle, (t, -)) &= \Diamond^-\langle \mathfrak{B}, B, \uparrow\rangle\\
\delta(\langle \mathfrak{B}, B, L\rangle, (t, -)) &= \Diamond\langle \mathfrak{B}, B, L\rangle\\
\delta(\langle \mathfrak{B}, B, L\rangle, \circ) &= \Diamond\langle \mathfrak{B}, B, L\rangle\\
\delta(\langle \mathfrak{B}, B, L\rangle, (t, \mathsf{tgt})) &= \langle \mathfrak{B}, B\rangle\\
\delta(\langle \mathfrak{B}, B, \uparrow\rangle, \circ) &= \Diamond^-\langle \mathfrak{B}, B, \uparrow\rangle\\
\delta(\langle \mathfrak{B}, B, \uparrow\rangle, (t, M)) &= \langle \mathfrak{B}, B\rangle
\end{aligned}
$$

$$
\begin{aligned}
\delta(\langle \mathfrak{B}, B, r^-\rangle, (t, M)) &= \Diamond\langle \mathfrak{B}, B, \downarrow\rangle && \text{if } M \notin \{\mathsf{tgt}, \mathsf{loop}\}\\
\delta(\langle \mathfrak{B}, B, r^-\rangle, (t, \mathsf{tgt})) &= \Diamond^-\langle \mathfrak{B}, B, R\rangle \vee \Diamond\langle \mathfrak{B}, B, \downarrow\rangle &&\\
\delta(\langle \mathfrak{B}, B, r^-\rangle, (t, \mathsf{loop})) &= \langle \mathfrak{B}, B\rangle \vee \Diamond\langle \mathfrak{B}, B, \downarrow\rangle &&\\
\delta(\langle \mathfrak{B}, B, R\rangle, (t, \mathsf{src})) &= \langle \mathfrak{B}, B\rangle &&
\end{aligned}
$$

$$\delta(\langle \mathfrak{B}, B, R\rangle, \sigma) = \Diamond^- \langle \mathfrak{B}, B, R\rangle \qquad \text{for all } \sigma \neq (t, \mathsf{src})$$
$$\delta(\langle \mathfrak{B}, B, \downarrow\rangle, (t, M)) = \langle \mathfrak{B}, B\rangle$$
$$\delta(\langle \mathfrak{B}, B, \downarrow\rangle, \circ) = \Diamond \langle \mathfrak{B}, B, \downarrow\rangle$$

The transitions for states $\langle \mathfrak{B}, \overline{B}\rangle$ are complementary. Correctness of the automaton is not difficult to show. In particular, using the acceptance condition, it is ensured that states of the form $\langle \mathfrak{B}, B\rangle$ do not occur infinitely often along a run. Thus, every $\exists \alpha.B$ is eventually witnessed. □

Proofs for Section 5

Lemma 6. *Let $\mathfrak{A} = (Q, \Sigma', q_0, \delta, \rho)$ be an NPTA and let $\Sigma_{\mathsf{fin}} = \{\sigma_1, \dots, \sigma_k\}$ and Σ_{inf} be disjoint subsets of Σ'. Then there is a family $(\varphi_S)_{S \in \mathcal{S}_\mathfrak{A}}$ of formulas of existential Presburger arithmetic such that for every $\mathbf{a} = (a_1, \dots, a_k) \in \mathbb{N}^k$, the following are equivalent:*

1. *$\mathbf{a} \in \mathsf{Mod}(\varphi_S)$ for some $S \in \mathcal{S}_\mathfrak{A}$;*
2. *there is a $\tau \in L(\mathfrak{A})$ such that*
   (a) *$\#_{\sigma_i}(\tau) = a_i$ for $1 \leq i \leq k$ and*
   (b) *$\#_\sigma(\tau) = \infty$ for every $\sigma \in \Sigma_{\mathsf{inf}}$.*

Given $\mathfrak{A}$, Σ_{inf}, Σ_{fin}, and an $S \in \mathcal{S}_\mathfrak{A}$, φ_S can be constructed in polynomial time.

Proof. Let $S = \{(q_1, \sigma_1, \Psi_1), \dots, (q_m, \sigma_m, \Psi_m)\} \in \mathcal{S}_\mathfrak{A}$. In order to define the φ_S, we consider a decomposition of the trees in $L(\mathfrak{A})$ into an initial part which contains all labels from Σ_{fin}, and in (several other) parts that contain no labels from Σ_{fin}. We start with defining an NTA that describes possible initial parts.

Define $Z = \{(q, \sigma) \mid (q, \sigma, \Psi) \in S \text{ for some } \Psi\}$. The NTA $\mathfrak{A}' = (Q, \Sigma \cup Z, q_0, \Delta')$ is obtained from $\mathfrak{A} = (Q, \Sigma, q_0, \Delta, \rho)$ by taking

$$\Delta' = \Delta \cup \{(q, (q, \sigma), \varepsilon) \mid (q, \sigma) \in Z\}.$$

Intuitively, $\mathfrak{A}'$ recognizes possible finite prefixes of elements from $L(\mathfrak{A})$ in which the leaves are additionally labeled with the states in which $\mathfrak{A}$ can visit the node in a successful run. This is realized by the definition of Δ' and the fact that Z is read off from the $\mathfrak{A}$-obligation set S. Formally, $L(\mathfrak{A}')$ consists of all finite $\Sigma \cup Z$-labeled trees (T, τ) such that there is some tree $(T', \tau') \in L(\mathfrak{A})$ which satisfies the following conditions:

- $T \subseteq T'$,
- τ' and τ coincide on all inner nodes of T, and
- there is an accepting run r of $\mathfrak{A}$ on τ' such that every leaf n of T is labeled with $(r(n), \tau'(n))$.

Let G be $\mathfrak{A}$' viewed as a CFG, c.f. Section 4, and $\varphi_G(\mathbf{x})$ be the Presburger formula that exists due to Theorem 2, where $\mathbf{x}$ is the sequence containing all x_σ with $\sigma \in \Sigma \cup Z$. The required formula φ_S has free variables y_i, for every

$\sigma_i \in \Sigma_{\mathsf{fin}}$, so we need to relate those y_i to the variables in $\mathbf{x}$. Moreover, we need to reflect the multiplicities in S. More precisely, if there are several $\mathfrak{A}$-obligations with the same pair (q, σ), this has to be reflected in φ_S. We denote with $k_{q,\sigma}$ the number of triples $(q_i, \sigma_i, \Psi_i) \in S$ such that $(q_i, \sigma_i) = (q, \sigma)$ and $\Psi_i \neq \emptyset$. The required formula φ_S is then defined as follows:

$$\varphi_S(y_1, \dots, y_k) = \exists \mathbf{x} \Big(\varphi_G(\mathbf{x}) \wedge \bigwedge_{(q,\sigma) \in Z} x_{(q,\sigma)} \geq k_{q,\sigma} \wedge \tag{$*$}$$

$$\bigwedge_{i=1}^{k} y_i = x_{\sigma_i} + \sum_{(q,\sigma_i) \in Z} x_{(q,\sigma_i)} \Big). \tag{$**$}$$

To finish the proof, we have to verify the equivalence of Points 1 and 2 in the lemma.

($1 \Rightarrow 2$). Let $\mathbf{a} \in \mathsf{Mod}(\varphi_S)$ for some $S \in \mathcal{S}_{\mathfrak{A}}$. That is, there is some $\mathbf{b} \in \mathsf{Mod}(\varphi_G)$ such that $(*)$ and $(**)$ are satisfied under the assignment of $y_1, \dots, y_k$ to $\mathbf{a}$ and $\mathbf{x}$ to $\mathbf{b}$. By definition of φ_G, there is a $\Sigma \cup Z$-labeled tree $\tau' \in L(\mathfrak{A}')$ such that $\#_\sigma(\tau') = b_\sigma$, for all $\sigma \in \Sigma \cup Z$. We construct a Σ-labeled tree τ as follows.

By definition of an $\mathfrak{A}$-obligation, there are $\Sigma \setminus \Sigma_{\mathsf{fin}}$-labeled trees $\tau_1, \dots, \tau_m$ such that for every $i \in \{1, \dots, m\}$ we have $\tau_i \in L(\mathfrak{A}^{q_i})$ and $\tau_i(\varepsilon) = \sigma_i$. By $(*)$ and the definition of $k_{q,\sigma}$, we can choose for every leaf n of τ' a tree $t(n)$ from $\tau_1, \dots, \tau_m$, such that each τ_i with $\Psi_i \neq \emptyset$ appears at least once. Obtain τ by replacing each leaf n with $t(n)$. Point 2 is now a consequence of the construction of τ and the relationship of the y_i and $\mathbf{x}$ enforced by $(**)$.

($2 \Rightarrow 1$). Assume some $\tau \in L(\mathfrak{A})$ such that $\#_\sigma(\tau) = \infty$ for every $\sigma \in \Sigma_{\mathsf{inf}}$ and $\#_{\sigma_i}(\tau) = a_i$ for every $\sigma_i \in \Sigma_{\mathsf{fin}}$. We show that $\mathbf{a} = (a_1, \dots, a_k) \in \mathsf{Mod}(\varphi_S)$ for some $S \in \mathcal{S}_{\mathfrak{A}}$.

Since $\tau \in L(\mathfrak{A})$, we can fix an accepting run r for τ. Observe that there is a depth ℓ such that below ℓ no symbol from Σ_{fin} occurs in τ. Moreover, there are nodes $n_1, \dots, n_k$ in level ℓ and a partition $\Sigma_1, \dots, \Sigma_k$ of Σ_{inf} such that, for all $i \in \{1, \dots, k\}$ and all $\sigma \in \Sigma_i$, we have $\#_\sigma(\tau_{n_i}) = \infty$. Define S by taking

$$S = \{(r(n_i), \tau(n_i), \Sigma_i) \mid i \in \{1, \dots, k\}\} \cup \{(r(n), \tau(n), \emptyset) \mid n \text{ in level } \ell\}.$$

By definition, all triples in S are $\mathfrak{A}$-obligations and thus S is an $\mathfrak{A}$-obligation set. Recall that Z is the set of all (q, σ) such that S contains a triple (q, σ, Ψ) for some Ψ. To show that $\mathbf{a}$ is a model for φ_S, consider the finite $\Sigma \cup Z$-labeled tree τ' defined as follows:

- $\tau'(n) = \tau(n)$, for every node $n \in T$ on a level smaller than ℓ;
- $\tau'(n) = (r(n), \tau(n))$, for every node $n \in T$ on level ℓ;
- there are no nodes on a level greater than ℓ.

By construction, $\tau' \in L(\mathfrak{A}')$. By Theorem 2, there is some $\mathbf{b} \in \mathsf{Mod}(\varphi_G)$ with $\mathbf{b}$ the sequence of all b_σ, $\sigma \in \Sigma \cup Z$, such that $\#_\sigma(\tau') = b_\sigma$, for all $\sigma \in \Sigma \cup Z$. Now,

equations $(*)$ and $(**)$ are satisfied by the definition of S and the construction of τ'.

It remains to note that φ_S can obviously be computed in polynomial time in the size of $\mathfrak{A}$, Σ_{inf}, Σ_{fin}, and S. □

Lemma 7. *Given a triple $(q, \sigma, \Psi) \in Q \times \Sigma' \times 2^{\Sigma_{\mathsf{inf}}}$, it is decidable in* NP *whether (q, σ, Ψ) is an $\mathfrak{A}$-obligation.*

In order to prove Lemma 7, we need the following auxiliary lemma, which uses *non-deterministic Büchi tree automata (NBTA)*. An NBTA is an NPTA where the priority function satisfies $\rho(q) \in \{0, 1\}$, for all $q \in Q$.

Lemma 8. *For every set $S_0 \subseteq \Sigma$, there is a class $\mathbf{A}$ of NBTA such that:*

1. *For every Σ-labeled tree τ such that $\#_\sigma(\tau) = \infty$, for all $\sigma \in S_0$, we have $\tau \in L(\mathfrak{A})$ for some $\mathfrak{A} \in \mathbf{A}$.*
2. *For every $\mathfrak{A} \in \mathbf{A}$ and $\tau \in L(\mathfrak{A})$, we have $\#_\sigma(\tau) = \infty$, for all $\sigma \in S_0$.*

Moreover, there is a non-deterministic polynomial-time Turing machine which generates $\mathbf{A}$.

Proof. The definition of the NBTAs in $\mathbf{A}$ is based on the following observation.

Claim 1. Given a set of labels $S \subseteq \Sigma$, a Σ-labeled tree (T, τ), and a node $n \in T$. The labels in S occur infinitely often in T_n iff one of the following is the case:

(A) there is a node $n' \in T_n$ and a partition $S_1 \uplus S_2$ of S such that the labels in S_i occur infinitely often in the i-th successor of n';
(B) there is an infinite path $n_0, n_1, \ldots$ with $n_0 = n$ such that, for every $\sigma \in S$, there are infinitely many i such that some node in T_{n_i} is labeled with σ.

Proof of Claim 1. The "if"-direction is trivial. For the "only-if"-direction, assume that all labels in S occur infinitely below n, but (A) is not satisfied. We construct the infinite path in (B) inductively. Start with $n_0 = 0$. For the induction step, let S_j, $j \in \{1, 2\}$ be the set of labels that occur infinitely often in the j-th successor of n_i. Since (A) is not satisfied, one of S_1 or S_2 is empty and the other one is S. If S_1 is empty, n_{i+1} is the second successor of n_i; if S_2 is empty, n_{i+1} is the first successor of n_i. Consequently, for every i we have that (i) all symbols in S occur infinitely often in T_{n_i}, and (ii) if n_{i+1} is the first (resp., second) successor of n_i, then all labels from S occur only finitely often below the second (resp., first) successor of n_i. Condition (B) is an immediate consequence of (i) and (ii). This finishes the proof of the Claim.

Intuitively, every NBTA in $\mathbf{A}$ implements a sequence of partitionings of Σ according to (A), and stops partitioning at some point. From then on, it continues by verifying the existence of the infinite path described in (B). Note that it is not difficult to construct a non-deterministic Büchi tree automaton $\mathfrak{A}_S$ (of size polynomial in S) that checks Condition (B) for S, that is, $\mathfrak{A}_S$ accepts a tree iff there is a path as described in (B); we omit the details.

Formally, the class $\mathbf{A}$ contains an NBTA $\mathfrak{A}_\tau$ for every finite, binary, 2^{S_0}-labeled tree (T, τ) which satisfies the following properties:

- $\tau(\varepsilon) = S_0$;
- for every $n \in T$ with successors n_1, n_2, $\tau(n_1) \uplus \tau(n_2)$ is a partition of $\tau(n)$.

Given such a tree (T, τ), $\mathfrak{A}_\tau$ is defined as follows. Its set of states consists of T and, for every leaf node n, the states from $\mathfrak{A}_{\tau(n)}$ with the initial state renamed to n. The initial state of $\mathfrak{A}_\tau$ is the root of τ. The transition relation is the union of all the transition relations of the Büchi automata associated to the leaves, and the following transitions, for every inner node $n \in T$ with successors n_1, n_2, and all $\sigma \in \Sigma$:

$$(n, \sigma, n_1 n_2), \quad (n, \sigma, n\emptyset), \quad (n, \sigma, \emptyset n), \quad (\emptyset, \sigma, \emptyset\emptyset), \quad (\emptyset, \sigma, \varepsilon).$$

The priority function assigns $\rho(n) = 1$ for all inner nodes $n \in T$. Thus, it is ensured that eventually a state corresponding to a leaf is reached. For the other states q which all occur in some $\mathfrak{A}_S$ with priority function ρ_S, we set $\rho(q) = \rho_S(q)$.

Claim 2. The class $\mathbf{A}$ of NBTAs satisfies Conditions 1 and 2 from the Lemma.

Proof of Claim 2. For Condition 2, let $\mathfrak{A}_{\hat{\tau}} \in \mathbf{A}$ and $\tau \in L(\mathfrak{A}_{\hat{\tau}})$, that is, there is an accepting run of $\mathfrak{A}_{\hat{\tau}}$ on τ. The priority function ensures that, on every infinite path, eventually a state n for some leaf node $n \in \hat{T}$ is reached. From there on, Condition (B) for $\hat{\tau}(n)$ is verified, that is, in T_n all the symbols in $\hat{\tau}(n)$ occur infinitely often. It remains to note that, by construction the labels in the leaves of $\hat{\tau}$ form a partition of Σ_{fin}.

For Condition 1, let τ be a Σ-labeled tree such that $\#_\sigma(\tau) = \infty$, for every $\sigma \in S_0$. We construct a (2^{S_0})-labeled tree $\hat{\tau}$ and an accepting run r of $\mathfrak{A}_{\hat{\tau}}$ on τ as follows. Start with a single node tree $\hat{\tau}$ with $\hat{\tau}(\varepsilon) = \Sigma_{\mathsf{fin}}$ and $r(\varepsilon) = \Sigma_{\mathsf{fin}}$. Then apply the following rule exhaustively:

- Choose a leaf node n in r and let its label be $r(n) = S$.
  - If S and n satisfy Condition (A) above, let n' be the witnessing node with successors n_1, n_2, and $S_1 \uplus S_2$ the witnessing partition of S. Set $r(m) = S$ for every m on the path from n to n', and $r(m) = \emptyset$ for every m that is a successor node of some node on the path from n to n', but not on the path itself. Moreover, set $r(n_1) = S_1$ and $r(n_2) = S_2$ and add in $\hat{\tau}$ two new children to the node labeled with S, and label them with S_1 and S_2.
  - Otherwise, by Claim 1, S and n satisfy Condition (B) above. By definition, the NBTA $\mathfrak{A}_S$ has an accepting run r' on the subtree starting with n. Then complete the run r by setting $r(n') = r'(n')$, for every $n' \in T_n$, $n \neq n'$.

It is routine to verify that the resulting run r is an accepting run of the constructed $\mathfrak{A}_{\hat{\tau}}$ on τ. This finishes the proof of the Claim. □

To finish the proof of Lemma 7, let us first argue that the NPTA $\mathfrak{A}$ needs only constantly many priorities, that is, we can assume that $\rho(q) \leq k$ for all $q \in Q$, for some constant k. Indeed, the 2APTA we start with is Büchi, that

is, needs only two priorities $0, 1$, and the transformation of 2APTAs to NPTAs increases the number of priorities linearly (in the original number of priorities). Thus, both $\mathfrak{A}$ and all the NPTAs in $\mathbf{A}$ have constantly many priorities. We can then proceed as follows:

1. Guess $\mathfrak{A}_1 \in \mathbf{A}$;
2. construct Muller tree automata $\mathfrak{A}'$, $\mathfrak{A}'_1$ equivalent to $\mathfrak{A}$, $\mathfrak{A}_1$, respectively;
3. compute the Muller tree automaton $\mathfrak{B}$ as the intersection of $\mathfrak{A}'$ and $\mathfrak{A}'_1$;
4. convert $\mathfrak{B}$ back to an NPTA $\mathfrak{B}'$;
5. check emptiness of $\mathfrak{B}$'. Return "yes" if $L(\mathfrak{B}')$ is non-empty; and "no" otherwise.

Note that the first step is (non-deterministic) polynomial time due to Lemma 8. Step 2 is polynomial because every NPTA can be transformed into a Muller automaton in polynomial time. Step 3 is polynomial since intersection of Muller automata can be computed in polynomial time. Step 4 is polynomial since $\mathfrak{B}$ has constantly many priorities. Finally, Step 5 is in NP by classic results on the non-emptiness problem for NPTAs. Correctness follows from Properties 1 and 2 given in Lemma 8.