Presburger Büchi Tree Automata with Applications to Logics with Expressive Counting

Bartosz Bednarczyk[1,2] , **Oskar Fiuk**[2]

[1]Computational Logic Group, Technische Universität Dresden, Germany
[2]University of Wrocław, Faculty of Mathematics and Computer Science
bartosz.bednarczyk@cs.uni.wroc.pl, 307023@uwr.edu.pl

Abstract

We introduce two versions of Presburger Automata with the Büchi acceptance condition, working over infinite, finite-branching trees. These automata, in addition to the classical ones, allow nodes for checking linear inequalities over labels of their children. We establish tight NP and EXPTIME bounds on the complexity of the non-emptiness problem for the presented machines. We demonstrate the usefulness of our automata models by polynomially encoding known DLs with Presburger constraints and inverses, improving the existing triply-exponential upper bound a single exponential.

1 Introduction

Description Logics (DLs) (Baader et al. 2017) are a robust family of first-order logic (FO) fragments designed for knowledge representation and reasoning. They serve as the core of the Web Ontology Languages (OWL) and are successfully used in the bio-medical domain, *e.g.* in Snomed CT (El-Sappagh et al. 2018). Unfortunately, the expressiveness of DLs is quite limited: they are not able to express even simple statistical properties. Such information is crucial to reason about real-life scenarios as witnessed by:

Example 1.1. *Consider a knowledge base of Citizens voting for their new mayor with the role votedFor. To ensure that every Citizen has exactly one vote, we need to express that votedFor$^-$ is functional. This situation can be modelled with:*

$$\mathit{func}(\mathit{votedFor}), \quad \mathit{Mayor} \sqsubseteq (>50\%)\mathit{votedFor}^-.\mathit{Citizen},$$

so (s)he received more than 50% of votes from the Citizens.

The above-mentioned properties can be formalised in extensions of the standard description logic $\mathcal{ALC}$ with Presburger arithmetic, as was done *e.g.* in (Demri and Lugiez 2010; Kupke and Pattinson 2010; Baader 2017). However, to model Example 1.1 one additionally requires the use of inverses of roles, making the logics from the cited papers not directly applicable in our setting. The decidability status of $\mathcal{ALC}$ with inverses and arithmetics was established only a few months ago in (Bednarczyk et al. 2021), but with a non-optimal 3NEXPTIME complexity. The aim of this short paper is to improve this complexity to a single exponential.

Our results. We follow the automata-theoretic approach to decidability of description logics, as treated in several papers, see e.g. (Baader, Hladik, and Peñaloza 2008; Calvanese, Carbotta, and Ortiz 2011; Calvanese, Eiter, and Ortiz 2014). We present two models of tree automata, working over infinite finite-branching trees, with Büchi accepting conditions. The first of them, called SPBTA, assumes the powerset alphabet and has a EXPTIME-complete non-emptiness problem. That makes SPBTA an attractive automata model for encoding modal and description logics with Presburger constraints. The other one, called PBTA, works over the usual alphabet and turns out to be NP-complete, complementing already existing results on finite tree automata with expressive counting from (Schwentick 2004; Seidl, Schwentick, and Muscholl 2008). As an application of our results, we provide a translation from an extension of $\mathcal{ALCI}$ with expressive cardinality constraints to our automata, establishing its EXPTIME-completeness.

2 Preliminaries

We assume familiarity with basics on formal languages, computability (Sipser 1996), and tree automata (Comon 1997). Let $\underline{n}$ denote the set $\{k \in \mathbb{N} \mid k < n\}$. By a *tree*, we mean a finite-branching tree, in which every branch is infinite. More formally, t is a tree if there is a positive $n \in \mathbb{N}$, so that t is a prefix-closed subset of $\underline{n}^\omega$ and for all $x \in \mathsf{t}$ we have $x0 \in \mathsf{t}$. The elements of t are called *nodes*. W.l.o.g. we assume that trees are *children-closed*, *i.e.* if $x \in \mathsf{t}$ and $n \in \mathbb{N}$ then $xn \in \mathsf{t}$ implies xk for all $k \in \underline{n}$. Given $x \in \mathsf{t}$ we define the subtree of x (resp. children of x), denoted $\mathsf{Subt}(x)$ (resp. $\mathsf{Chld}(x)$), as the set $\{z \in \mathsf{t} \mid z = xy\}$ (resp. $\{xn \in \mathsf{t} \mid n \in \mathbb{N}\}$). The *degree* $\mathsf{d}(x)$ (resp. *height* $\mathsf{h}(x)$) of x is $|\mathsf{Chld}(x)|$ (resp. $|x|$). A *branch* of t is any infinite sequence $v_0, v_1, v_2, \ldots$ of nodes of t, such that $v_0 = \varepsilon$ and for all i the node v_{i+1} is a children of v_i. A Σ-labelled tree is a pair (ℓ, t), composed of a tree t and a mapping $\ell : \mathsf{t} \to \Sigma$. Sometimes we will also think about *finite* trees, defined in a natural way, with all the definitions adopted to them as the reader may expect.

By a *linear constraint* we mean an expression of the form $C + \Sigma_{i=0}^{n} a_i x_i \bowtie \Sigma_{i=0}^{m} b_i y_i + D$, where x_i, y_i are variables from some countably-infinite set $\mathcal{V}$, $\bowtie$ is one of $\leq, <, \equiv_k$ (the last one denotes equality modulo[1] k) and a_i, b_i, k, C, D are integers encoded in binary and $(k > 0)$. We define *systems of constraints* and their *solutions over* $\mathbb{N}$ in the expected

[1]Constraints with $\equiv_k$ can be eliminated (Bednarczyk 2020).

way. Given a solution $S : \mathcal{V} \to \mathbb{N}$ to a system E, the *support* $\mathsf{supp}(S)$ of S is defined as the set $\{v \in \mathcal{V} \mid S(v) \neq 0\}$. Testing *solvability* (over $\mathbb{N}$) of systems of constraints is NP-complete (Borosh, Flahive, and Treybig 1986). Moreover, the following less-known lemma provides the existence of a "sparse and small" solution; for more details consult: (Eisenbrand and Shmonin 2006, Theorem 1), (Baader 2017, Lemma 3) and (Papadimitriou 1981).

Lemma 2.1. *Let $\mathcal{I}$ be a system of E linear constraints with integer coefficients absolutely bounded by C. Assume a solution S to $\mathcal{I}$ (over $\mathbb{N}$). Then there is a solution S' to $\mathcal{I}$ (over $\mathbb{N}$) with $|\mathsf{supp}(S')| \leq 2E \cdot \log(4EC)$ and $\mathsf{supp}(S') \subseteq \mathsf{supp}(S)$. Moreover, the maximal absolute value assigned to variables by S' is bounded by $V(E \cdot C)^{2E+1}$.*

3 Suitable tree automata models

Henceforth, we consider two tree automata models, starting from the more succinct and expressive one:

Definition 3.1. *A Succinct Presburger Büchi Tree Automata (SPBTA) $\mathcal{A}$ is a tuple $\mathcal{A} = (\Sigma, Q, F, Q_0, I)$, where all of its components are finite, Σ is an* alphabet*, Q is a set of* states*, $F \subseteq Q$ is a set of* final states*, $Q_0 \subseteq Q$ is an* initial configuration *and I is a finite set of* instructions *of the form:*

- $\pm q \to \pm q'$, $q \wedge q' \to q''$ *for* $q, q', q'' \in Q$,
- $q \to \pm a, \pm a \to q$ *for* $q \in Q$ *and* $a \in \Sigma$, *and*
- $q \to \mathcal{C}, \mathcal{C} \to q$, *where $\mathcal{C}$ is a linear constraint over* $\mathcal{V} = \{x_S \mid S \subseteq Q\}$. *For succinctness, only the mentioned variables from $\mathcal{V}$ are required to be explicitly written in $\mathcal{C}$.*

Our automata works over 2^Σ-labelled trees and labels each node with a subset of Q. Such a labelling is required to be compatible with Q_0, *i.e.* ε is labelled with Q_0. Moreover, it should be compatible with I, *i.e.* that the presence of a state in a node v implies the absence/presence of a state/letter in v, and that linear constraints over the labellings of v's children, where the value of the variable x_S is the total number of v's children labelled by some superset of S. Formally:

Definition 3.2. *A* run *of an SPBTA $\mathcal{A} = (\Sigma, Q, F, Q_0, I)$ over a 2^Σ-labelled tree $(\ell_\mathsf{t}, \mathsf{t})$ is a 2^Q-labelled tree $(\ell^\mathcal{A}_\mathsf{t}, \mathsf{t})$ such that $\ell^\mathcal{A}_\mathsf{t}(\varepsilon) = Q_0$, and for all $v \in \mathsf{t}$ and $q \in \ell^\mathcal{A}_\mathsf{t}(v)$:*

- *if $(q \to q') \in I$ (resp. $(q \to -q') \in I$) with $q' \in Q$, then $q' \in \ell^\mathcal{A}_\mathsf{t}(v)$ (resp. $q' \notin \ell^\mathcal{A}_\mathsf{t}(v)$), and if $(q \wedge q') \to q'' \in I$ with $q', q'' \in Q$ then $q, q' \in \ell^\mathcal{A}_\mathsf{t}(v)$ implies $q'' \in \ell^\mathcal{A}_\mathsf{t}(v)$,*
- *if $(q \to \pm a) \in I$ (resp. $(q \to -a) \in I$) with $a \in \Sigma$, then $a \in \ell_\mathsf{t}(v)$ (resp. $a \notin \ell_\mathsf{t}(v)$),*
- *if $(q \to \mathcal{C}) \in I$ with $\mathcal{C}$ being a linear constraint, then $x_S \mapsto |\{u \in \mathsf{Chld}(v) : S \subseteq \ell^\mathcal{A}_\mathsf{t}(u)\}|$ is a solution to $\mathcal{C}$.*

Omitted cases are symmetric. A run is successful*, if every branch of t has inf. many elements v fulfilling the condition: "there is an accepting state $q_f \in F$ such that $q_f \in \ell_\mathcal{A}(v)$".*

An SPBTA $\mathcal{A}$ accepts *a 2^Σ-labelled tree $(\ell_\mathsf{t}, \mathsf{t})$ if there is a successful run over $(\ell_\mathsf{t}, \mathsf{t})$. A language of $\mathcal{A}$, denoted $\mathcal{L}(\mathcal{A})$, is the set of all 2^Σ-labelled trees accepted by $\mathcal{A}$.*

The succinct representation of the alphabet and constraints in SPBTA will turn out to be especially useful for encoding models of modal and description logics with Presburger constraints (due to their tree-like model property). In what follows we present yet another model of tree automata, which is simpler due to the use of a standard alphabet. Within a run, nodes are labelled with exactly one state, and therefore, many of the presented instructions can be removed.

Definition 3.3. *A Presburger Büchi Tree Automata (PBTA) $\mathcal{A}$ is a tuple $\mathcal{A} = (\Sigma, Q, F, q_0, I)$, where all its components are finite, Σ is an* alphabet*, Q is a set of* states*, $F \subseteq Q$ is a set of* final states*, $q_0 \in Q$ is an* initial configuration *and I is a finite set of* instructions *of the form:*

- *$(q, a) \to \mathcal{C}$, where $q \in Q$, $a \in \Sigma$ and $\mathcal{C}$ is a linear constraint with variables $\mathcal{V} = \{x_q \mid q \in Q\}$.*

We define runs of PBTAs analogously to the case of SPBTAs.

Definition 3.4. *A* run *of a PBTA $\mathcal{A} = (\Sigma, Q, F, q_0, I)$ over a Σ-labelled tree $(\ell_\mathsf{t}, \mathsf{t})$ is a Q-labelled tree $(\ell^\mathcal{A}_\mathsf{t}, \mathsf{t})$ such that $\ell^\mathcal{A}_\mathsf{t}(\varepsilon) = q_0$, and for all $v \in \mathsf{t}$ with $q_v{=}\ell^\mathcal{A}_\mathsf{t}(v)$ and $a_v{=}\ell_\mathsf{t}(v)$:*

- *if $((q_v, a_v) \to \mathcal{C}) \in I$, then $x_q \mapsto |\{u \in \mathsf{Chld}(v) \mid q = \ell^\mathcal{A}_\mathsf{t}(u)\}|$ is a solution to $\mathcal{C}$.*

We say that a run is successful*, if every branch of t has infinitely many elements v fulfilling the condition: "there is an accepting state $q_f \in F$ such that $q_f = \ell_\mathcal{A}(v)$". The notion of a tree* accepted *and the* language of *PBTA are defined analogously to the case of SPBTA.*

The *non-emptiness* problem asks whether the language of a given (S)PBTA is non-empty. An immediate reduction from the solvability (over $\mathbb{N}$) of systems of linear equations yields NP-hardness for testing non-emptiness of PBTA.

Lemma 3.5. *The non-emptiness for PBTAs is* NP*-hard.*

Proof. Let $\mathcal{I}$ be a system of linear equations, and let $\Sigma = \{a_0\}, Q = F = \{q_v \mid v \in \mathcal{V}(\mathcal{I})\}$, $I = \{(q_v, a_0) \to \mathcal{C} \mid v \in \mathcal{V}(\mathcal{I}), \mathcal{C} \in \mathcal{I}\}$. Choose v_0 to be an arbitrary variable in $\mathcal{V}(\mathcal{I})$. Then the language recognized by automaton $\mathcal{A} = (\Sigma, Q, F, q_{v_0}, I)$ is non-empty iff $\mathcal{I}$ has a solution. □

In the sequel, we provide a matching upper bound.

4 Non-emptiness problem for PBTA is in NP

Henceforth, we fix a PBTA $\mathcal{A} = (\Sigma, Q, F, q_0, I)$. By its size, denoted with $|\mathcal{A}|$, we mean the size of some reasonable succinct representation of $\mathcal{A}$, where the numbers in constraints are encoded in *binary*. We start with auxiliary definitions.

Definition 4.1. *Let $(\ell_\mathcal{A}, \mathsf{t})$ be an accepting run of $\mathcal{A}$ on a tree $(\ell_\mathsf{t}, \mathsf{t})$. Then a state $q \in Q$ has* an occurrence in a node *$v \in \mathsf{t}$ iff $\ell_\mathcal{A}(v) = q$. If there exists a node $v \in \mathsf{t}$, such that q occurs in v, then q has* an occurrence in an automaton run *$(\ell_\mathcal{A}, \mathsf{t})$. An occurrence of $q \in Q$ in $v \in \mathsf{t}$ is* safe *iff either $q \in F$ or on each branch $v_1, v_2, v_3, \ldots$ starting in v there exists an occurrence of some state from F before the next occurrence of q (if any),* i.e. $\inf\{i \mid \ell_\mathcal{A}(v_i) \in F\} < \inf\{i \mid \ell_\mathcal{A}(v_i) = q\}$. *A* safe subtree *of a safe occurrence $q \in Q$ in $v \in \mathsf{t}$, denoted as $\mathsf{SSubt}(q, v)$, is a pair (v, t') such that t' is a singleton $\{\varepsilon\}$ whenever $q \in F$ or otherwise is a finite tree $\{x \mid vx \in \mathsf{Subt}(v), \forall st = x, t \neq \varepsilon,\ \ell_\mathcal{A}(vs) \notin F\}$,* i.e. *accepting states occur only in leaves.*

Lemma 4.2. *If $q \in Q$ has an occurrence in an accepting run $(\ell_{\mathcal{A}}, \mathfrak{t})$ of $\mathcal{A}$, then it has a safe one.*

Proof. Suppose that there is $q \in \ell_{\mathcal{A}}(\mathfrak{t})$, such that no occurrence is safe. Clearly, $q \notin F$. Then one can construct an infinite branch without any state from F, yielding a contradiction with the Büchi acceptance condition. Employing the following process. Start with any occurrence of q in some $v \in \mathfrak{t}$. Then by assumption it is not safe. Therefore, it contains a branch that reaches another occurrence of q in some other v' before reaching any state from F. Repeat from v'. □

To get an NP-upper bound, we construct a *regular tree*, generated by a pair of functions and a state, that will serve as a witness for the non-emptiness test of a given automaton.

Definition 4.3. *For functions $\alpha : Q \to \Sigma$, $\gamma : Q \to Q^+$, and an element $q_0 \in Q$, define inductively a family of sets:*

- $T_0 = \{(\varepsilon, q_0)\}$, *and*
- $T_{n+1} = \{(xk, \gamma(q)_k) \mid (x, q) \in T_n, k = 1, \ldots, |\gamma(q)|\}$.

Let $\mathcal{T} = \bigcup_{n \geq 0} T_n$. Then

- $\mathfrak{t} = \{x \mid (x, q) \in \mathcal{T}\}$ *is a tree,*
- ℓ_Q *is a Q-labeling of* $\mathfrak{t}$ *such that* $(x, \ell_Q(x)) \in \mathcal{T}$,
- ℓ_Σ *is a Σ-labeling of* $\mathfrak{t}$ *such that* $\ell_\Sigma(x) = \alpha(\ell_Q(x))$.

The role of α is to name elements with letters and the role of γ is to create fresh state-labelled children of a given node. Note that we are working with unordered trees, thus w.l.o.g. we assume that γ is "sorted". So γ can represented as a function $\Gamma : Q \to \mathbb{N}^Q$ defined as:

$$\gamma(q) = \underbrace{q_1 \ldots q_1}_{\Gamma(q)(q_1) \text{ times}} \quad \cdots \quad \underbrace{q_{|Q|} \ldots q_{|Q|}}_{\Gamma(q)(q_{|Q|}) \text{ times}} .$$

The role of Γ is to encode potential solutions to a system of inequalities from the I component of $\mathcal{A}$. Finally, we say that $(\ell_Q, \mathfrak{t})$ (resp. $(\ell_\Sigma, \mathfrak{t})$) is *$Q$-generated* (resp. *$\Sigma$-generated*) tree by a triple (q_0, α, Γ).

Lemma 4.4. *Let $u = (q_0, \alpha, \Gamma)$ generate a tree $\mathfrak{t}$ accepted by $\mathcal{A}$. Then there is a triple $u' = (q_0, \alpha, \Gamma')$ generating another tree $\mathfrak{t}'$ that is still accepted by $\mathcal{A}$, and which has small values in Γ,* i.e. $\log \max\{\Gamma(q)(q') \mid q, q' \in Q^2\} = O(poly(|\mathcal{A}|))$.

Proof. Since u generates $\mathfrak{t}$ accepted by $\mathcal{A}$, the values of $\Gamma(q)$ (after a renaming) are solutions to constraints $q \to \mathcal{C}$ given by $\mathcal{A}$. By Lemma 2.1 we get smaller solutions that can be encoded as Γ'. Then the tree $\mathfrak{t}'$ generated by (q_0, α, Γ') is clearly accepted by $\mathcal{A}$. Indeed, suppose that there is a branch with finitely many accepting states. Then, by the inclusion of supports for the solutions and due to the way the tree $\mathfrak{t}'$ is generated, there will be a branch in the original tree $\mathfrak{t}$ with the same Q-labelling, yielding a contradiction. □

The next crucial lemma shows how to generate a small regular tree, serving as a witness for our non-emptiness test.

Lemma 4.5. *If $\mathcal{L}(\mathcal{A})$ is nonempty, then there is a tree $\mathfrak{t}$ and a triple $u = (q_0, \alpha, \Gamma)$ satisfying: (1) $(\ell_\Sigma, \mathfrak{t}) \in \mathcal{L}(\mathcal{A})$, and is Σ-generated by u, (2) $(\ell_Q, \mathfrak{t})$ is an accepting run of $\mathcal{A}$ on the tree $(\ell_\Sigma, \mathfrak{t})$, and is Q-generated by u, and (3) the triple u has size polynomial in $|\mathcal{A}|$.*

Proof. Fix a tree $(\ell_{\mathfrak{t}'}, \mathfrak{t}') \in \mathcal{L}(\mathcal{A})$ and a corresponding accepting run $(\ell^{\mathcal{A}}_{\mathfrak{t}'}, \mathfrak{t}')$ of $\mathcal{A}$. Let $Q_0 = \{q_1, \ldots, q_k\}$ be the set of states appearing in $\ell^{\mathcal{A}}_{\mathfrak{t}'}(\mathfrak{t}')$. For each $q_i \in Q_0$ fix a safe occurrence in some $v_i \in \mathfrak{t}'$ and let $(u_i, \mathfrak{t}_i) = \mathsf{SSubt}(q_i, v_i)$. We also fix some post-order enumeration of nodes from $\mathfrak{t}_i$ and let $\mathtt{PO}(\mathfrak{t}_i, v)$ denotes the position of $v \in \mathfrak{t}_i$ in such order.

A *solution* in a node v is a function $S_v : Q \to \mathbb{N}$ such that

$$S_v(q) = |\{u \in \mathsf{Chld}(v) : \ell^{\mathcal{A}}_{\mathfrak{t}'}(u) = q\}|.$$

We define a *witness* injection $j : Q_0 \to \mathbb{N}^2 \times \mathbb{N}^*$, that will be used to define α and Γ components of u, as follows:

$$j(q_i) = \min\{(r, s, x) \mid x \in \mathfrak{t}_r, q_i = \ell^{\mathcal{A}}_{\mathfrak{t}'}(u_r x), \mathtt{PO}(\mathfrak{t}_r, x) = s\},$$

where the minimum is taken with respect to the lexicographic order. The intuition behind the function j is that it assigns, for each state, the deepest leftmost occurrence of this state in the safe subtree with the lowest possible index. Therefore, if the node is not a leaf, any outgoing edge leads to a child with a strictly smaller value of j. Since all leaves are labelled with accepting states, any branch in the resulting tree that visits the same node twice must pass through the accepting state.

Suppose that $j(q_i) = (r_i, s_i, x_i)$. Then we simply put $\alpha(q_i) = \ell_{\mathfrak{t}'}(u_{r_i} x_i)$ and $\Gamma(q_i) = S_{u_{r_i} x_i}$ for each $q_i \in Q_0$, and let $\alpha(q) = \bot$ and $\Gamma(q) = \bot$ for each $q \in Q \setminus Q_0$. One can show (see Appendix A) that $(\ell_Q, \mathfrak{t})$ is indeed an accepting run of $\mathcal{A}$ on the tree $(\ell_\Sigma, \mathfrak{t})$ with respect to the given definition of (q_0, α, Γ). Thus, by applying Lemma 4.4 we conclude that u has a polynomial size description. □

Lemma 4.6. *Given $\mathcal{A}$ and a tuple $u = (q_0, \alpha, \Gamma)$, it is in* PTIME *to test if u generates a tree from $\mathcal{L}(\mathcal{A})$.*

Proof. We can clearly verify all linear constraints from the I component of $\mathcal{A}$, since Γ encodes all the required solutions. For the Büchi acceptance condition build a directed graph on the vertices Q and edges $E = \{(q, q') \in Q^2 \mid q \notin F, \Gamma(q)(q') > 0\}$. Then if the resulting graph (Q, E) is acyclic, output `Yes`, and `No` otherwise. □

Thus as a corollary of Lemmas 3.5, 4.5, 4.6 we conclude:

Theorem 1. *The non-emptiness problem for Presburger Büchi Tree Automata is* NP*-complete.*

5 Non-emptiness for SPBTA

Fix an SPBTA $\mathcal{A} = (\Sigma, Q, F, Q_0, I)$. Let N be the number of constraints in I and let C be the minimal absolute bound on the coefficients from I. We start with a "small accepted tree lemma":

Lemma 5.1. *If there is an accepting run $(\ell^{\mathcal{A}}_{\mathfrak{t}}, \mathfrak{t})$ of $\mathcal{A}$ on some tree $(\ell_{\mathfrak{t}}, \mathfrak{t})$, then there is a tree $(\ell_{\mathfrak{t}'}, \mathfrak{t}')$ with an accepting run $(\ell^{\mathcal{A}}_{\mathfrak{t}'}, \mathfrak{t}')$ of $\mathcal{A}$ satisfying the following conditions:*

- *for all nodes $v \in \mathfrak{t}'$ we have $|\mathsf{Chld}(v)| \leq 2^{|Q|}(N \cdot C)^{2N+1}$ and $|\{\ell^{\mathcal{A}}_{\mathfrak{t}'}(u) \mid u \in \mathsf{Chld}(v)\}| \leq 2N \cdot \log(4CN)$.*
- *On any path $v_1, v_2, \ldots, v_n$ of length $n > 2^{|Q|}$ in $\mathfrak{t}'$ there is a node v such that $\ell^{\mathcal{A}}_{\mathfrak{t}'}(v)$ contains a final state.*

Proof. We construct $(\ell^{\mathcal{A}}_{\mathfrak{t}'}, \ell_{\mathfrak{t}'}, \mathfrak{t}')$ recursively, starting from $(\ell^{\mathcal{A}}_{\mathfrak{t}}, \ell_{\mathfrak{t}}, \mathfrak{t})$. The second condition can be established simply by taking any finite path $v_1, v_2, \ldots, v_n$ of length $n > 2^{|Q|}$ violating it and by observing, by the pigeonhole principle, that two nodes v_i, v_j $(i < j)$ have the same value of $\ell^{\mathcal{A}}_{\mathfrak{t}'}$. Hence, we obtain a new tree by replacing the subtree of v_i by a subtree of v_j. We repeat the process indefinitely. For the first condition, assume that there is a node v violating the condition. Per each subset $X \subseteq Q$ let v_X be any fixed child of v with $\ell^{\mathcal{A}}_{\mathfrak{t}'}(v_X) = X$ (if it exists). Let $\mathcal{I} = \{\mathcal{C} \mid q \in \ell^{\mathcal{A}}_{\mathfrak{t}'}(v), (q \to \mathcal{C}) \in I\}$ be a system of inequalities. Note that in $\mathcal{I}$ we have unknowns of the form x_X per each $X \subseteq Q$, which by the semantics of $\mathcal{A}$ counts nodes labelled by some superset of X. But $\mathcal{I}$ can be rewritten into a system $\mathcal{I}'$ (by applying the inclusion–exclusion principle) so that the unknowns x_X indicates the total number of children labelled by *precisely the set* X. Thus, the assignment $S : X \mapsto |\{u \in \mathsf{Chld}(v) \mid \ell^{\mathcal{A}}_{\mathfrak{t}'}(u) = X\}|$ is the solution to $\mathcal{I}'$. By employing Lemma 2.1 to $\mathcal{I}'$ we infer the existence of a "sparse and small" solution S' to $\mathcal{I}'$ with $\mathsf{supp}(S') \subseteq \mathsf{supp}(S)$. We conclude by modifying $\mathfrak{t}'$ as follows: we remove all descendants of v and for any subset $X \subseteq Q$ with $S'(x_X) \neq 0$ we assign as children of v precisely $S'(x_X)$ copies of the previously selected elements v_X. The labellings $\ell^{\mathcal{A}}_{\mathfrak{t}'}, \ell_{\mathfrak{t}'}$ are as indicated by the "copies". The resulting tree is accepted by $\mathcal{A}$: systems of constraints are satisfied (again by applying the inclusion-exclusion principle on S') and the Büchi acceptance condition still holds, since the "copied" subtrees were present in the original run. □

We next describe APSPACE (= EXPTIME by (Chandra and Stockmeyer 1976)) procedure for testing $\mathcal{L}(\mathcal{A}) \neq \emptyset$. It will be tableaux-like: we start by guessing a node, then we guess its children, verify the consistency of the guess with the transition function of $\mathcal{A}$ and repeat the process from a universally selected child. In the pseudocode below we identify a node with a pair $(q, \ell) \in 2^Q \times 2^\Sigma$. We employ two counters, stored in binary. The counter `FCnt` is responsible for counting how many times we encountered an accepting state (if $\texttt{FCnt} > 2^{|Q|}$ we know that we ended up in the same accepting configuration twice and hence, we can build the tree by making precisely the same choices as we did in the past). The counter `QCnt` is responsible for counting how many configurations we have visited without entering into an accepting state. If $\texttt{QCnt} > 2^{|Q|}$ then some non-final state was surely repeated twice, violating the second condition of Lemma 5.1.

1 **Guess** the root node $v = (q, \ell)$ and set $\texttt{FCnt} = 0$.
2 **If** $\texttt{FCnt} > 2^{|Q|}$ **then** *Accept* **else** set $\texttt{QCnt} = 0$.
3 **Guess** $\leq 2N \cdot \log(4CN)$ nodes (q_i, ℓ_i) that will be repeated **guessed** $n_i \leq 2^{|Q|}(N \cdot C)^{2N+1}$ times.
4 Verify whether the guessed children are consistent with the transition function of $\mathcal{A}$.
5 **Universally choose** (q_i, ℓ_i) as $v = (q, \ell)$.
6 **If** q contains a final state **then** `FCnt++` and **goto** 2.
7 `QCnt++`. **If** $\texttt{QCnt} > 2^{|Q|}$ **then** *Reject* **else goto** 3.

A correctness proof can be found in Appendix B. We get:

Theorem 2. *Non-emptiness for SPBTAs is in* EXPTIME.

6 Applications in Logic

We assume familiarity with description logics (DLs) (Baader et al. 2017). An $\mathcal{ALCISCC}$ ontology[2] is composed of the following concept equivalences:

$$A \equiv \neg B, \quad A \equiv B \sqcap B', \quad A \equiv C + (\Sigma_{i=1}^n \lambda_i \# s_i.B_i) \bowtie D,$$

where A, B, B_i are concept names, C, D, λ_i are integers, $\bowtie$ is one of $<, \leq, \geq, >, \equiv_k$ with $k \in \mathbb{N}_+$, and s_i is a (possibly inverted) role name.The semantics of $\mathcal{ALCISCC}$ is defined via (possibly infinite) finite-branching interpretations. The novelty is that $d \in (C + (\Sigma_{i=1}^n \lambda_i \# s_i.B_i) \bowtie D)^{\mathcal{I}}$ iff

$$C + (\Sigma_{i=1}^n \lambda_i \cdot |\{e \in \Delta^{\mathcal{I}} \mid (d, e) \in s_i^{\mathcal{I}} \wedge e \in B_i^{\mathcal{I}}\}|) \bowtie D \text{ holds.}$$

Take any $\mathcal{ALCISCC}$-ontology $\mathcal{O}$. We can construct an equisatisfiable ontology $\mathcal{O}'$ by introducing fresh role name *extra* and inserting $(\top \equiv \#\mathit{extra}.\top > 0)$ inside $\mathcal{O}$. Hence, by applying standard notions of unravellings (Rudolph 2011, Sec. 6.5), one can show that that every satisfiable $\mathcal{O}'$ has a *tree model*, *i.e.* a model whose domain is a tree and the interpretation of roles is restricted to parent-child and child-parent pairs. Such models will be encoded with a help of SBPTAs.

By a *tree encoding* of an *tree model* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of $\mathcal{I}$ we mean a Σ-labelled tree $(\ell_{\mathfrak{t}}, \mathfrak{t} = \Delta^{\mathcal{I}})$, such that:

- Σ is composed of all concept names, role names and inverted role names appearing in $\mathcal{O}'$,
- if $v \in A^{\mathcal{I}}$ iff $A \in \ell_{\mathfrak{t}}(v)$,
- $(v, vk) \in r^{\mathcal{I}}$ (resp. $(v, vk) \in r^{\mathcal{I}}$) for some $v \in \Delta$ and its child vk, iff r (resp. r^-) belongs to $\ell_{\mathfrak{t}}(vk)$.

For the opposite way, note that a Σ-labelled tree $(\ell_{\mathfrak{t}}, \mathfrak{t})$ is an encoding of some tree model of $\mathcal{O}'$ whenever:

- For all $A \equiv \neg B \in \mathcal{O}'$ we have that all nodes $v \in \mathfrak{t}$ satisfy $A \in \ell_{\mathfrak{t}}(v)$ iff $B \notin \ell_{\mathfrak{t}}(v)$,
- For all $A \equiv B \sqcap B' \in \mathcal{O}'$ we have that all nodes $v \in \mathfrak{t}$ satisfy $A \in \ell_{\mathfrak{t}}(v)$ iff $B, B' \in \ell_{\mathfrak{t}}(v)$,
- For all $(A \equiv C + (\Sigma_{i=1}^n \lambda_i \# s_i.B_i) \bowtie D) \in \mathcal{O}'$ we have that for all nodes v we have the equivalence $A \in \ell_{\mathfrak{t}}(v)$ iff $C + (\Sigma_{i=1}^n \lambda_i | vk \in \mathsf{Chld}(v) : s_i, B_i \in \ell_{\mathfrak{t}}(vk)|) \bowtie D$.

Hence, it suffices to construct a SPBTA $\mathcal{A}$ accepting Σ-labelled tree satisfying the above requirements. In the construction $\mathcal{A}$ it suffices to introduce states $q_a, q_{a \wedge b}, q_I$ per each names $a, b \in \Sigma$ and constraints $I = C + (\Sigma_{i=1}^n \lambda_i \# s_i.B_i) \bowtie D$ appearing in $\mathcal{O}'$. We adjust their intended meaning by requiring $\mathcal{A}$ to contain the instructions $q_a \to a, q_{a \wedge b} \to q_a, q_{a \wedge b} \to q_b$ as well as $q_I \to C + (\Sigma_{i=1}^n \lambda_i x_{\{q_{s_i}, q_{B_i}\}}) \bowtie D$ (plus the reversed implications). Having this said, ensuring that a tree $\mathfrak{t} \in \mathcal{L}(\mathcal{A})$ encodes a tree model of $\mathcal{O}'$ by enforcing additional constraints on $\mathcal{A}$ is straightforward. Our encoding is polynomial, thus by EXPTIME-hardness of $\mathcal{ALC}$ we get:

Corollary 6.1. *The satisfiability problem for $\mathcal{ALCISCC}$ over finite-branching models is* EXPTIME*-complete. Hence, the non-emptiness problem for SPBTA is* EXPTIME*-hard.*

[2] Due to space limits, we simplified a quite complicated definition from (Baader 2017), but the expressivity stays the same.

Acknowledgements

This work was supported by the Polish Ministry of Science and Higher Education program "Diamentowy Grant" no. DI2017 006447. Results from this paper will appear in the BSc thesis of Oskar Fiuk, written under informal supervision of Bartosz Bednarczyk at the University of Wrocław. We are grateful to Tim Lyon for proofreading and language improvements.

References

Baader, F.; Horrocks, I.; Lutz, C.; and Sattler, U. 2017. *An Introduction to Description Logic*. Cambridge University Press.

Baader, F.; Hladik, J.; and Peñaloza, R. 2008. Automata can show pspace results for description logics. *Inf. Comput.* 206(9-10):1045–1056.

Baader, F. 2017. A new description logic with set constraints and cardinality constraints on role successors. In *FroCoS 2017*, volume 10483, 43–59. Springer.

Bednarczyk, B.; Orlowska, M.; Pacanowska, A.; and Tan, T. 2021. On Classical Decidable Logics Extended with Percentage Quantifiers and Arithmetics. In *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2021, December 15-17, 2021, Virtual Conference*, volume 213 of *LIPIcs*, 36:1–36:15.

Bednarczyk, B. 2020. One-variable logic meets presburger arithmetic. *Theor. Comput. Sci.* 802:141–146.

Borosh, I.; Flahive, M.; and Treybig, B. 1986. Small solutions of linear diophantine equations. *Discret. Math.* 58(3):215–220.

Calvanese, D.; Carbotta, D.; and Ortiz, M. 2011. A practical automata-based technique for reasoning in expressive description logics. In Walsh, T., ed., *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, 798–804. IJCAI/AAAI.

Calvanese, D.; Eiter, T.; and Ortiz, M. 2014. Answering regular path queries in expressive description logics via alternating tree-automata. *Inf. Comput.* 237:12–55.

Chandra, A. K., and Stockmeyer, L. J. 1976. Alternation. In *17th Annual Symposium on Foundations of Computer Science, Houston, Texas, USA, 25-27 October 1976*, 98–108. IEEE Computer Society.

Comon, H. 1997. Tree automata techniques and applications.

Demri, S., and Lugiez, D. 2010. Complexity of modal logics with presburger constraints. *J. Appl. Log.*

Eisenbrand, F., and Shmonin, G. 2006. Carathéodory bounds for integer cones. *Oper. Res. Lett.* 34(5):564–568.

El-Sappagh, S.; Franda, F.; Ali, F.; and Kwak, K.-S. 2018. Snomed ct standard ontology based on the ontology for general medical science. *BMC medical informatics and decision making* 18(1):1–19.

Kupke, C., and Pattinson, D. 2010. On modal logics of linear inequalities. In Beklemishev, L. D.; Goranko, V.; and Shehtman, V. B., eds., *AIML*.

Papadimitriou, C. H. 1981. On the complexity of integer programming. *Journal of the ACM (JACM)* 28(4):765–768.

Rudolph, S. 2011. Foundations of description logics. In *Reasoning Web International Summer School*, 76–136. Springer.

Schwentick, T. 2004. Trees, automata and XML. In Beeri, C., and Deutsch, A., eds., *Proceedings of the Twenty-third ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 14-16, 2004, Paris, France*, 222. ACM.

Seidl, H.; Schwentick, T.; and Muscholl, A. 2008. Counting in trees. In Flum, J.; Grädel, E.; and Wilke, T., eds., *Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas]*, volume 2 of *Texts in Logic and Games*, 575–612. Amsterdam University Press.

Sipser, M. 1996. Introduction to the theory of computation. *ACM Sigact News* 27(1):27–29.