First Steps Towards Taming Description Logics with Strings

Stéphane Demri, Karin Quaas

First Steps Towards Taming Description Logics with Strings*

Stéphane Demri[1] and Karin Quaas[2]

[1]Université Paris-Saclay, CNRS, ENS Paris-Saclay,
Laboratoire Méthodes Formelles, 91190, Gif-Sur-Yvette, France
[2] Universität Leipzig, Fakultät für Mathematik und Informatik

Abstract. We consider the description logic $\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma)$ over the concrete domain $\mathcal{D}_\Sigma = (\Sigma^*, \prec, =, (=_{\mathfrak{w}})_{\mathfrak{w}\in\Sigma^*})$, where $\prec$ is the strict prefix order over finite strings in Σ^*. Using an automata-based approach, we show that the concept satisfiability problem w.r.t. general TBoxes for $\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma)$ is ExpTime-complete for all finite alphabets Σ. As far as we know, this is the first complexity result for an expressive description logic with a nontrivial concrete domain on strings.

1 Introduction

Description logics with concrete domains. A concrete domain is a relational structure with a fixed nonempty domain and a family of relations. In this paper, we are most and for all interested in the concrete domain $(\Sigma^*, \prec, =, (=_{\mathfrak{w}})_{\mathfrak{w}\in\Sigma^*})$, where Σ is a finite alphabet, $\prec$ is the strict prefix relation over Σ^*, $=$ is the usual equality relation, and $=_{\mathfrak{w}}$ stands for equality with $\mathfrak{w}$. Other typical examples of concrete domains (also playing a role herein) are $(\mathbb{N}, <, =, (=_z)_{z\in\mathbb{N}})$ and $(\mathbb{Z}, <, =, (=_z)_{z\in\mathbb{Z}})$, that are the (nonnegative) integers with the usual order relation $<$, equality, and equality with z.

We aim to reason about concrete domains using description logics. A standard way to do so is to enrich the semantical structures with values from a concrete domain (see e.g. [5,14,22]); then, specific atomic concepts are used to express constraints between these values. In description logics with concrete domains, the domain elements are enriched with tuples of values coming from the concrete domain, see e.g. [5,30,31,32,33,7]. Constraints on concrete domains embedded in concepts from description logics may quickly be expressive enough to encode counting mechanisms, leading to the undecidability of the main reasoning tasks, see e.g. [30]. However, nontrivial properties of concrete domains have been identified to get decidability, see e.g. [11,33,9] and also [14].

String theories. Description logics with concrete domains on strings are often evoked in the literature, see e.g. [5,10,4,24], but such logics are seldom studied. There are a few exceptions, see e.g. [23] handling strings with equality and inequality relations (only). In a way, string domains remain only a potentiality for description logics with concrete domains, although it is believed that concrete domains on strings could be useful in ontologies. Reasoning about strings

* The second author is supported by the Deutsche Forschungsgemeinschaft (DFG), project 504343613.

is often required in program verification, and much effort has been dedicated towards designing solvers that handle string theories, see e.g. [28,1,2]. An explanation for the lack of works on description logics may be the complexity inherent to string theories. For instance, first-order theory on strings with concatenation is undecidable [37]. On the other hand, satisfiability of word equations is in PSpace [34,36,20]. Herein, we are interested in the challenging question of deciding reasoning tasks for description logics with a non-trivial string domain.

Our motivations. What is particularly interesting about concrete domains on strings is to observe that these domains are absolutely not captured by the recent and sophisticated methods for determining decidability of description logics with concrete domains, see e.g. [33,8,9,39]. Moreover, the string domain $(\Sigma^*, \prec, =, (=_{\mathfrak{w}})_{\mathfrak{w}\in\Sigma^*})$, in the following denoted by $\mathcal{D}_\Sigma$, is known to be difficult to handle, see e.g. [13, Theorem 1]. This applies also to the concrete domain $\mathbb{N}$ (which can be understood as the string domain $\mathcal{D}_\Sigma$ with a singleton alphabet), but for this one, the remarkable works [15,27,26] lead to the ExpTime-completeness of reasoning tasks for the description logic $\mathcal{ALCF}^{\mathcal{P}}(\mathbb{Z})$. The concrete domain $\mathbb{N}$ still requires complex developments, but at least it is known today how to manage it, see e.g. [18,40,14]. Our motivation in this work is to investigate the decidability/complexity status of $\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma)$, that is, for the nontrivial string domain with the prefix order. To do so, we take advantage of recent results on tree constraint automata on $\mathbb{Z}$ from [19] combined with an encoding of string constraints by numerical constraints from [17]. These are only some first steps to tame reasoning tasks for description logics with string domains, and of course, other string domains are possibly interesting, see e.g. [35].

Our contributions. In Section 2, we introduce the description logic $\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma)$, similarly to the definition of $\mathcal{ALCF}^{\mathcal{P}}(\mathbb{Z})$ in [27,26]. In Section 3, we introduce the class of tree constraint automata (TCA) accepting infinite finite-branching trees with nodes labelled by letters from a finite alphabet and finite tuples of values in Σ^*. Our definition for TCA naturally extends the constraint automata for words (see e.g. [16,38,21,40,25,43]) as well as a similar one for trees on $(\mathbb{Z}, <, =)$ from [19, Section 3.1]. In Section 4, the nonemptiness problem for TCA is shown ExpTime-complete. Though ExpTime-hardness is a consequence of [19, Section 3.1], the ExpTime-membership is by reduction to the nonemptiness problem for TCA on $\mathbb{N}$ by lifting arguments from [17, Section 3] to the automata-based setting.

In Section 5, we show how to reduce the concept satisfiability problem w.r.t. general TBoxes for $\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma)$ (written $\mathrm{TSAT}(\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma))$) to the nonemptiness problem for TCA, following the automata-based approach developed in [42] (see also [44,3]). To do so, in Section 2, we establish a simple form for $\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma)$ concepts from which TCA are defined, adapting the developments from [15, Lemma 15] and [27, Lemma 5]. Though we use a standard approach in Section 5, we need to carefully handle the constraints in the TCA in order to provide a complexity analysis that leads to the optimal upper bound. The complexity of $\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma)$ concepts requires sophisticated TCA constructions and involved developments. As a result, we establish that $\mathrm{TSAT}(\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma))$ is ExpTime-

complete for all finite alphabets Σ. As far as we know, this is the first complexity characterisation for a reasoning task related to a description logic with a non-trivial string domain. As explained above, we reuse or adapt several results from the literature (not always related to description logics), and we provide several new insights to combine them adequately.

2 Description Logics with String Domains

Given a finite alphabet Σ, we consider the concrete domain $\mathcal{D}_\Sigma \stackrel{\text{def}}{=} (\Sigma^*, \prec, =, (=_{\mathfrak{w}})_{\mathfrak{w}\in\Sigma^*})$, where $\prec$ is the strict prefix relation on Σ^*, $=$ is the equality on Σ^*, and $=_{\mathfrak{w}}$ is a unary predicate stating the equality with the string $\mathfrak{w}$. In the following, we use $\text{card}(\Sigma)$ to denote the cardinality of Σ.

Let $\text{VAR} = \{\mathtt{x}, \mathtt{y}, \ldots\}$ be a countably infinite set of variables (also called *registers* in [27] and *concrete features* in the description logic literature). A *term* $\mathtt{t}$ *over* VAR is an expression of the form $S^i\mathtt{x}$, where $\mathtt{x} \in \text{VAR}$ and S^i is a (possibly empty) sequence of i symbols 'S'. A term $S^i\mathtt{x}$ should be understood as a variable (that needs to be interpreted) but, later on, we will see that the prefix S^i will have a relational interpretation. We write T_{VAR} to denote the set of all terms over VAR. For all $i \in \mathbb{N}$, we write $\text{T}^{\leqslant i}_{\text{VAR}}$ to denote the subset of terms of the form $S^j\mathtt{x}$, where $j \leqslant i$. For instance, $\text{T}^{\leqslant 0}_{\text{VAR}} = \text{VAR}$. An *atomic constraint* θ *over* T_{VAR} is an expression of one of the forms below:

$$\mathtt{t} \prec \mathtt{t}' \qquad \mathtt{t} = \mathtt{t}' \qquad =_{\mathfrak{w}} (\mathtt{t}) \text{ (also written } \mathtt{t} = \mathfrak{w}),$$

where $\mathfrak{w} \in \Sigma^*$ and $\mathtt{t}, \mathtt{t}' \in \text{T}_{\text{VAR}}$. A *constraint* Θ is defined as a Boolean combination of atomic constraints. Constraints are interpreted on valuations $\mathfrak{v} : \text{T}_{\text{VAR}} \to \Sigma^*$ that assign elements from Σ^* to the terms in T_{VAR}, so that $\mathfrak{v}$ *satisfies* θ, written $\mathfrak{v} \models \theta$ iff the interpretation of the terms in θ makes θ true in Σ^* in the usual way. Boolean connectives are interpreted as usual. A constraint Θ is *satisfiable* $\stackrel{\text{def}}{\Leftrightarrow}$ there is a valuation $\mathfrak{v} : \text{T}_{\text{VAR}} \to \Sigma^*$ such that $\mathfrak{v} \models \Theta$.

Below, we define the description logic $\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma)$ (over the concrete domain $\mathcal{D}_\Sigma$) defined exactly as the description logic $\mathcal{ALCF}^{\mathcal{P}}(\mathcal{Z}_c)$ from [27] except that $\mathcal{Z}_c$ is replaced by $\mathcal{D}_\Sigma$. We deliberately use the notations from [27] whenever possible and we provide a formal definition for $\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma)$ to be self-contained. Let $\mathsf{N_C} = \{A, B, \ldots\}$ and $\mathsf{N_R} = \{r, s, \ldots\}$, respectively, be countably infinite sets of *concept names* and *role names*. We further assume that $\mathsf{N_R}$ contains a subfamily $\mathsf{N_F} \subseteq \mathsf{N_R}$ of *functional role names* (a.k.a abstract features). A *role path* $P = r_1 \cdots r_n$ is a (possibly empty) word in $\mathsf{N}^*_{\mathbf{R}}$. We use $|P|$ to denote the length of P (possibly zero). The set of $\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma)$-*concepts* is defined as follows.

$$C ::= \top \mid \bot \mid A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \exists r.C \mid \forall r.C \mid \exists P.[\![\Theta]\!] \mid \forall P.[\![\Theta]\!],$$

where $A \in \mathsf{N_C}$, $r \in \mathsf{N_R}$, P is a role path, Θ is a Boolean constraint in Σ^* built over terms of the form $S^j\mathtt{x}$. Moreover, if $S^j\mathtt{x}$ occurs in Θ, then we require $j \leqslant |P|$. An *axiom* is an expression of the form $C \sqsubseteq D$, where C, D are $\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma)$ concepts. A *terminological box* $\mathcal{T}$ *(TBox, for short)* is a finite set of axioms.

An *interpretation* is a tuple $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}, \mathfrak{v})$, where $\Delta^{\mathcal{I}}$ is a nonempty set (the *domain*), $\mathfrak{v}\colon \Delta^{\mathcal{I}} \times \mathrm{VAR} \to \Sigma^*$ (the *valuation function*), and $\cdot^{\mathcal{I}}$ is an interpretation function that assigns $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ to every concept name $A \in \mathsf{N_C}$, and $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to every role name $r \in \mathsf{N_R}$. For all $f \in \mathsf{N_F}$, we require $\{(\mathfrak{a}, \mathfrak{a}'), (\mathfrak{a}, \mathfrak{a}'')\} \subseteq f^{\mathcal{I}}$ implies $\mathfrak{a}' = \mathfrak{a}''$, that is, $f^{\mathcal{I}}$ is a partial function. Given a role path $P = r_1 r_2 \dots r_n$, we define $P^{\mathcal{I}}$ to be the set of all tuples $(\mathfrak{a}_0, \dots, \mathfrak{a}_n) \in \Delta^{n+1}$ such that $(\mathfrak{a}_{i-1}, \mathfrak{a}_i) \in r_i^{\mathcal{I}}$ for all $i \in [1, n]$. Given an interpretation $\mathcal{I}$ and a tuple $\pi = (\mathfrak{a}_0, \mathfrak{a}_1, \dots, \mathfrak{a}_n)$ of elements in $\Delta^{\mathcal{I}}$, constraints Θ_1, Θ_2, and $\mathfrak{w} \in \Sigma^*$, we define

- $\mathcal{I}, \pi \models S^i\mathtt{x} \prec S^j\mathtt{y} \stackrel{\text{def}}{\Leftrightarrow} \mathfrak{v}(\mathfrak{a}_i, \mathtt{x}) \prec \mathfrak{v}(\mathfrak{a}_j, \mathtt{y})$,
- $\mathcal{I}, \pi \models S^i\mathtt{x} = S^j\mathtt{y} \stackrel{\text{def}}{\Leftrightarrow} \mathfrak{v}(\mathfrak{a}_i, \mathtt{x}) = \mathfrak{v}(\mathfrak{a}_j, \mathtt{y})$; $\mathcal{I}, \pi \models S^i\mathtt{x} = \mathfrak{w} \stackrel{\text{def}}{\Leftrightarrow} \mathfrak{v}(\mathfrak{a}_i, \mathtt{x}) = \mathfrak{w}$,
- $\mathcal{I}, \pi \models \neg\Theta_1 \stackrel{\text{def}}{\Leftrightarrow}$ not $\mathcal{I}, \pi \models \Theta_1$; $\mathcal{I}, \pi \models \Theta_1 \wedge \Theta_2 \stackrel{\text{def}}{\Leftrightarrow} \mathcal{I}, \pi \models \Theta_1$ and $\mathcal{I}, \pi \models \Theta_2$,
- $\mathcal{I}, \pi \models \Theta_1 \vee \Theta_2 \stackrel{\text{def}}{\Leftrightarrow} \mathcal{I}, \pi \models \Theta_1$ or $\mathcal{I}, \pi \models \Theta_2$.

We extend the interpretation function $\cdot^{\mathcal{I}}$ to complex concepts as follows:

- $\top^{\mathcal{I}} \stackrel{\text{def}}{=} \Delta^{\mathcal{I}}$, $\bot^{\mathcal{I}} \stackrel{\text{def}}{=} \varnothing$, $(\neg C)^{\mathcal{I}} \stackrel{\text{def}}{=} \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$,
- $(C \sqcap D)^{\mathcal{I}} \stackrel{\text{def}}{=} C^{\mathcal{I}} \cap D^{\mathcal{I}}$; $(C \sqcup D)^{\mathcal{I}} \stackrel{\text{def}}{=} C^{\mathcal{I}} \cup D^{\mathcal{I}}$,
- $(\exists r.C)^{\mathcal{I}} \stackrel{\text{def}}{=} \{\mathfrak{a} \in \Delta^{\mathcal{I}} \mid$ there is $\mathfrak{a}' \in \Delta^{\mathcal{I}}$ such that $\mathfrak{a}' \in C^{\mathcal{I}}$ and $(\mathfrak{a}, \mathfrak{a}') \in r^{\mathcal{I}}\}$,
- $(\forall r.C)^{\mathcal{I}} \stackrel{\text{def}}{=} \{\mathfrak{a} \in \Delta^{\mathcal{I}} \mid$ for all $\mathfrak{a}' \in \Delta^{\mathcal{I}}, (\mathfrak{a}, \mathfrak{a}') \in r^{\mathcal{I}}$ implies $\mathfrak{a}' \in C^{\mathcal{I}}\}$,
- $(\exists P.[\![\Theta]\!])^{\mathcal{I}} \stackrel{\text{def}}{=} \{\mathfrak{a}_0 \in \Delta^{\mathcal{I}} \mid$ there exist $\mathfrak{a}_1, \dots, \mathfrak{a}_n \in \Delta^{\mathcal{I}}$ s.t. $\pi = (\mathfrak{a}_0, \mathfrak{a}_1, \dots, \mathfrak{a}_n) \in P^{\mathcal{I}}$ and $\mathcal{I}, \pi \models \Theta\}$,
- $(\forall P.[\![\Theta]\!])^{\mathcal{I}} \stackrel{\text{def}}{=} \{\mathfrak{a}_0 \in \Delta^{\mathcal{I}} \mid$ for all $\mathfrak{a}_1, \dots, \mathfrak{a}_n \in \Delta^{\mathcal{I}}, \pi = (\mathfrak{a}_0, \mathfrak{a}_1, \dots, \mathfrak{a}_n) \in P^{\mathcal{I}}$ implies $\mathcal{I}, \pi \models \Theta\}$.

An interpretation $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$, written $\mathcal{I} \models \mathcal{T}$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all axioms $C \sqsubseteq D$ in $\mathcal{T}$. The *concept satisfiability problem with respect to general TBoxes*, written $\mathrm{TSAT}(\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma))$, is defined as follows:

Input: An $\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma)$ concept C_0, and a TBox $\mathcal{T}$.
Question: Is there an interpretation $\mathcal{I}$ of $\mathcal{T}$ such that $\mathcal{I} \models \mathcal{T}$ and $C_0^{\mathcal{I}} \neq \varnothing$?

For instance, $\exists rr'.[\![S^2\mathtt{x} \prec \mathtt{x}]\!]$, $\{\top \sqsubseteq \exists rr'.[\![S^2\mathtt{y} \prec \mathtt{x}]\!], \top \sqsubseteq \exists rr'.[\![\mathtt{x} \prec S^2\mathtt{x}]\!], \top \sqsubseteq \exists r.\top, \top \sqsubseteq \exists r'.\top\}$ is a positive instance of $\mathrm{TSAT}(\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma))$.

Given an $\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma)$ concept C_0 and a TBox $\mathcal{T}$, we write $\mathrm{sub}(C_0, \mathcal{T})$ to denote the set of subconcepts obtained from the concepts in C_0 and $\mathcal{T}$. A concept C is in *simple form* iff it is in negation normal form (negation occurs only in constraints) and terms are only from $\mathrm{T}^{\leqslant 1}_{\mathrm{VAR}}$. For instance, $\exists rr'.[\![S^2\mathtt{y} \prec \mathtt{x}]\!]$ is not in simple form but all the concepts in $\exists r.\exists r'.\exists\varepsilon.[\![\mathtt{y} \prec \mathtt{x}^{\dagger\dagger}]\!]$, $\top \sqsubseteq \forall r.[\![\mathtt{x} = S\mathtt{x}^{\dagger}]\!]$ and $\top \sqsubseteq \forall r'.[\![\mathtt{x}^{\dagger} = S\mathtt{x}^{\dagger\dagger}]\!]$ are. Negation normal form is easy to get by standard means as each concept constructor has its dual and the constraints Θ are closed under negations. In Section 5, we reduce $\mathrm{TSAT}(\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma))$ to the nonemptiness problem for *tree constraint automata* (defined in Section 3). For this, we assume that the input concept and the concepts occurring in the TBox are in simple form. In Proposition 1 we state that this assumption is without loss of generality and does not cause any computational harm.

Proposition 1. *Let C_0 be an $\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma)$ concept and $\mathcal{T}$ be a TBox. One can construct in polynomial-time in the size of $C_0, \mathcal{T}$ a concept C_0' and a finite TBox $\mathcal{T}'$ in simple form such that $C_0, \mathcal{T}$ is a positive instance of* $\mathrm{TSAT}(\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma))$ *iff $C_0', \mathcal{T}'$ is a positive instance of* $\mathrm{TSAT}(\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma))$.

Proposition 1 is analogous to [15, Lemma 15] and [27, Lemma 5]. Though based on similar principles, our proof is slightly simpler than the ones cited above, because we demand less from concepts in simple form, as the forthcoming tree constraint automata can handle such concepts (see Section 5).

Before defining tree constraint automata, we give a formal definition of *trees*. Given $d \geqslant 1$, a *labeled tree of degree d* is a map $\mathbf{t} : dom(\mathbf{t}) \to \mathbf{\Sigma}$ where $\mathbf{\Sigma}$ is some (potentially infinite) alphabet and $dom(\mathbf{t})$ is an infinite subset of $[0, d-1]^*$, that is, if $\mathbf{n} \cdot j \in dom(\mathbf{t})$ for some $\mathbf{n} \in [0, d-1]^*$ and $j \in [0, d-1]$, then $\mathbf{n} \in dom(\mathbf{t})$ and $\mathbf{n} \cdot i \in dom(\mathbf{t})$ for all $0 \leqslant i < j$, too. The elements of $dom(\mathbf{t})$ are called *nodes*. The empty word ε is the *root node* of $\mathbf{t}$. For every $\mathbf{n} \in dom(\mathbf{t})$, the elements $\mathbf{n} \cdot i$ (with $i \in [0, d-1]$) are called the *children nodes of* $\mathbf{n}$, and $\mathbf{n}$ is called the *parent node of* $\mathbf{n} \cdot i$. We say that the tree $\mathbf{t}$ is a *full d-ary tree* if every node $\mathbf{n}$ has exactly d children $\mathbf{n} \cdot 0, \ldots, \mathbf{n} \cdot (d-1)$. Given a tree $\mathbf{t}$ and a node $\mathbf{n}$ in $dom(\mathbf{t})$, an infinite *path* in $\mathbf{t}$ starting from $\mathbf{n}$ is an infinite sequence $\mathbf{n} \cdot j_1 \cdot j_2 \cdot j_3 \ldots$, where $j_i \in [0, d-1]$ and $\mathbf{n} \cdot j_1 \ldots j_i \in dom(\mathbf{t})$ for all $i \geqslant 1$.

3 Tree Constraint Automata Manipulating Strings

In this section, we introduce the class of tree constraint automata that accept sets of trees of the form $\mathbf{t} : [0, d-1]^* \to (\mathbf{\Sigma} \times (\Sigma^*)^\beta)$ for some finite alphabet $\mathbf{\Sigma}$ and some $\beta \geqslant 1$. Note that two alphabets are involved here: $\mathbf{\Sigma}$ is a finite alphabet as usually in automata, Σ is inherited from $\mathcal{D}_\Sigma$ and used to interpret β variables at each position of the trees. The transition relation of such automata states constraints between the β string values at a node and the string values at its children nodes. To do so, we write $\mathrm{TreeCons}(\beta)$ to denote the Boolean constraints built over the terms $\mathtt{x}_1, \ldots, \mathtt{x}_\beta, S\mathtt{x}_1, \ldots, S\mathtt{x}_\beta$. These constraints are used to define the transition relation of such automata. We also write $\mathtt{x}'_i$ to denote the term $S\mathtt{x}_i$, and we shall use valuations $\mathfrak{v}$ with profile $\{\mathtt{x}_i, \mathtt{x}'_i \mid i \in [1, \beta]\} \to \Sigma^*$. In the forthcoming definition, the acceptance condition on infinite branches is a Büchi condition, but this can be easily extended to more general conditions. Moreover, the definition is specific to the concrete domain Σ^* but it can be easily adapted to other concrete domains. A *tree constraint automaton* (TCA) on $\mathcal{D}_\Sigma$ is a tuple $\mathbb{A} = (Q, \mathbf{\Sigma}, d, \beta, Q_{\mathrm{in}}, \delta, F)$, where

- Q is a finite set of locations; $\mathbf{\Sigma}$ is a finite alphabet,
- $d \geqslant 1$ is the (branching) degree of (the trees accepted by) $\mathbb{A}$,
- $\beta \geqslant 1$ is the number of variables (a.k.a. registers) interpreted in Σ^*,
- $Q_{\mathrm{in}} \subseteq Q$ is the set of initial locations,
- δ is a finite subset of $Q \times \mathbf{\Sigma} \times (\mathrm{TreeCons}(\beta) \times Q)^d$, the transition relation. That is, δ consists of tuples of the form $(q, \mathtt{a}, (\Theta_0, q_0), \ldots, (\Theta_{d-1}, q_{d-1}))$, where $q, q_0, \ldots, q_{d-1} \in Q$, $\mathtt{a} \in \mathbf{\Sigma}$, and $\Theta_0, \ldots, \Theta_{d-1}$ are constraints built over $\mathtt{x}_1, \ldots, \mathtt{x}_\beta, \mathtt{x}'_1, \ldots, \mathtt{x}'_\beta$ for the concrete domain $\mathcal{D}_\Sigma$.

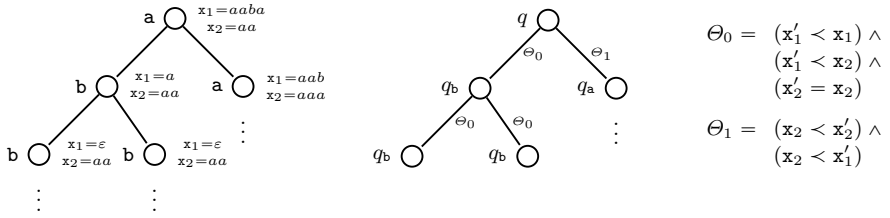

Fig. 1: On the left, the prefix of an infinite tree $\mathbf{t}$ with two string variables $\mathtt{x}_1$ and $\mathtt{x}_2$. In the middle, the beginning of a run of $\mathbb{A}$ from Example 1 on $\mathbf{t}$.

– $F \subseteq Q$ encodes the Büchi acceptance condition.

Let $\mathbf{t} : [0, d-1]^* \to (\mathbf{\Sigma} \times (\Sigma^*)^\beta)$ be an infinite full d-ary tree over $\mathbf{\Sigma} \times (\Sigma^*)^\beta$. A *run* of $\mathbb{A}$ on $\mathbf{t}$ is a mapping $\rho : [0, d-1]^* \to Q$ satisfying the following conditions:

– $\rho(\varepsilon) \in Q_{\mathrm{in}}$;
– for every $\mathbf{n} \in [0, d-1]^*$ with $\mathbf{t}(\mathbf{n}) = (\mathtt{a}, \boldsymbol{v})$ and $\rho(\mathbf{n}) = q$, $\mathbf{t}(\mathbf{n} \cdot i) = (\mathtt{a}_i, \boldsymbol{v}_i)$ and $\rho(\mathbf{n} \cdot i) = q_i$ for all $0 \leqslant i < d$, there exists $(q, \mathtt{a}, (\Theta_0, q_0), \ldots, (\Theta_{d-1}, q_{d-1})) \in \delta$ and $\Sigma^* \models \Theta_i(\boldsymbol{v}, \boldsymbol{v_i})$ for all $0 \leqslant i < d$. Here, $\Sigma^* \models \Theta_i(\boldsymbol{v}, \boldsymbol{v_i})$ is short for $[\vec{\mathtt{x}} \leftarrow \boldsymbol{v}, \vec{\mathtt{x}'} \leftarrow \boldsymbol{v_i}] \models \Theta_i$, where $[\vec{\mathtt{x}} \leftarrow \boldsymbol{v}, \vec{\mathtt{x}'} \leftarrow \boldsymbol{v_i}]$ is a valuation $\mathfrak{v}$ on $\{\mathtt{x}_j, \mathtt{x}'_j \mid j \in [1, \beta]\}$ with $\mathfrak{v}(\mathtt{x}_j) = \boldsymbol{v}(j)$ and $\mathfrak{v}(\mathtt{x}'_j) = \boldsymbol{v_i}(j)$ for all $j \in [1, \beta]$.

Note that string expressions labelling the transitions may state constraints between string values at a node and its children nodes.

Suppose ρ is a run of $\mathbb{A}$ on $\mathbf{t}$. Given an infinite path $\pi = j_1 \cdot j_2 \cdot j_3 \ldots$ in ρ starting from the root, we define $\inf(\rho, \pi)$ to be the set of control states that appear infinitely often in $\rho(\varepsilon)\rho(j_1)\rho(j_1 \cdot j_2)\rho(j_1 \cdot j_2 \cdot j_3) \ldots$. A run ρ is *accepting* if for all paths π in ρ starting from ε, we have $\inf(\rho, \pi) \cap F \neq \varnothing$. We write $\mathrm{L}(\mathbb{A})$ to denote the set of trees $\mathbf{t}$ that admit an accepting run.

Example 1. Let $\mathbb{A} = (\{q, q_{\mathtt{a}}, q_{\mathtt{b}}\}, \{\mathtt{a}, \mathtt{b}\}, 2, 2, \{q\}, \delta, \{q_{\mathtt{a}}, q_{\mathtt{b}}\})$, and δ containing precisely $(q, \mathtt{a}, (\Theta_0, q_{\mathtt{b}}), (\Theta_1, q_{\mathtt{a}}))$, $(q_{\mathtt{a}}, \mathtt{a}, (\top, q_{\mathtt{a}}), (\top, q_{\mathtt{a}}))$, and $(q_{\mathtt{b}}, \mathtt{b}, (\Theta_0, q_{\mathtt{b}}), (\Theta_0, q_{\mathtt{b}}))$, where $\Theta_0 = (\mathtt{x}'_1 \prec \mathtt{x}_1) \wedge (\mathtt{x}'_1 \prec \mathtt{x}_2) \wedge (\mathtt{x}'_2 = \mathtt{x}_2)$ and $\Theta_1 = (\mathtt{x}_2 \prec \mathtt{x}'_2) \wedge (\mathtt{x}_2 \prec \mathtt{x}'_1)$. In Figure 1, we show the beginning of a run on the tree $\mathbf{t}$ on the left. Note that this run cannot be extended to an infinite run of $\mathbb{A}$ on $\mathbf{t}$: on the leftmost branch, there is no value for $\mathtt{x}'_1$ that satisfies the constraint $\mathtt{x}'_1 \prec \mathtt{x}_1$ for the value of $\mathtt{x}_1$ being ε, hence no transition from $\mathbb{A}$ can be taken. In fact, there cannot be any infinite tree for which there exists some accepting run, and hence $\mathrm{L}(\mathbb{A}) = \varnothing$.

As usual, the *nonemptiness problem for TCA*, written NE(TCA), takes as input a TCA $\mathbb{A} = (Q, \mathbf{\Sigma}, d, \beta, Q_{\mathrm{in}}, \delta, F)$ and asks whether $\mathrm{L}(\mathbb{A})$ is nonempty. We aim to prove that this problem is ExpTime-complete. Unlike (plain) Büchi tree automata [42], the number of transitions in a tree constraint automaton is *a priori* unbounded (TreeCons(β) is infinite) and the maximal size of a constraint occurring in transitions is unbounded too. In particular, this means that the number

of transitions in δ, denoted by $\mathrm{card}(\delta)$, is a priori unbounded, even if Q and $\mathbf{\Sigma}$ are fixed. We write $\mathtt{MCS}(\mathbb{A})$ to denote the maximal size of a constraint occurring in $\mathbb{A}$. The complexity of the nonemptiness problem must therefore also take into account these parameters.

Below, we use TCA on the concrete domain $(\mathbb{N}, <, =, (=_n)_{n\in\mathbb{N}})$. These are defined as for $\mathcal{D}_\Sigma$, but with Σ being a singleton alphabet. Moreover, we assume that the natural numbers are encoded in binary. As a consequence of [19, Section 4], the nonemptiness problem for TCA on $\mathbb{N}$ is ExpTime-complete and the purpose of the next section is to show how to generalise this result for any concrete domain $\mathcal{D}_\Sigma$ (with a non-singleton alphabet Σ).

Our tree constraint automata differ from Presburger Büchi tree automata defined in [41,12] for which, in the runs, arithmetical expressions are constraints between the numbers of children labelled by different locations. Herein, the string expressions state constraints between string values (possibly at different nodes).

4 Nonemptiness Problem for TCA on $\mathcal{D}_\Sigma$

To reduce the nonemptiness problem for TCA on $\mathcal{D}_\Sigma$ to the nonemptiness problem for TCA on the concrete domain $(\mathbb{N}, <, =, (=_n)_{n\in\mathbb{N}})$, we show how to take advantage of [17, Lemma 6] dedicated to the transformation of prefix constraints into Boolean combinations of atomic constraints on $\mathbb{N}$. For the sake of being self-contained, we recall below a few definitions useful in Section 4.2.

4.1 From string constraints to natural number constraints

Given a string $\mathfrak{w} \in \Sigma^*$, we write $|\mathfrak{w}|$ to denote its *length*. Given $\mathfrak{w}, \mathfrak{w}' \in \Sigma^*$, we write $\mathit{clen}(\mathfrak{w}, \mathfrak{w}')$ to denote the *length of the longest common prefix* between $\mathfrak{w}$ and $\mathfrak{w}'$. We view the arguments of $\mathit{clen}(\cdot)$ as a set, so that $\mathit{clen}(\mathfrak{w}, \mathfrak{w}')$ and $\mathit{clen}(\mathfrak{w}', \mathfrak{w})$ are identical. More precisely, there are $\mathfrak{w}_0$, $\mathfrak{w}_1$, and $\mathfrak{w}'_1$ such that $\mathfrak{w} = \mathfrak{w}_0 \cdot \mathfrak{w}_1$, $\mathfrak{w}' = \mathfrak{w}_0 \cdot \mathfrak{w}'_1$ and, $\mathfrak{w}_1$ and $\mathfrak{w}'_1$ cannot start by the same first letter, if any. We set $\mathit{clen}(\mathfrak{w}, \mathfrak{w}') \stackrel{\text{def}}{=} |\mathfrak{w}_0|$. For example, $\mathit{clen}(aba, abbbab) = 2$. So, $\mathit{clen}(\mathfrak{w}, \mathfrak{w}) = |\mathfrak{w}|$, and $\mathfrak{w}$ is a strict prefix of $\mathfrak{w}'$ iff $\mathit{clen}(\mathfrak{w}, \mathfrak{w}) = \mathit{clen}(\mathfrak{w}, \mathfrak{w}')$ and $\mathit{clen}(\mathfrak{w}, \mathfrak{w}) < \mathit{clen}(\mathfrak{w}', \mathfrak{w}')$. Here are simple properties, see e.g. [17, Proposition 2], that play a special role in the sequel (assuming $\mathrm{card}(\Sigma) = k$).

(I) For all $\mathfrak{w}, \mathfrak{w}' \in \Sigma^*$, $|\mathfrak{w}| \geqslant \mathit{clen}(\mathfrak{w}, \mathfrak{w}')$.

(II) For all $\mathfrak{w}_0, \mathfrak{w}_1, \ldots, \mathfrak{w}_k \in \Sigma^*$ such that $\mathit{clen}(\mathfrak{w}_0, \mathfrak{w}_1) = \cdots = \mathit{clen}(\mathfrak{w}_0, \mathfrak{w}_k)$ and for all $i \in [0, k]$, $\mathit{clen}(\mathfrak{w}_0, \mathfrak{w}_1) < |\mathfrak{w}_i|$, there are $i \neq j \in [1, k]$ such that $\mathit{clen}(\mathfrak{w}_0, \mathfrak{w}_1) < \mathit{clen}(\mathfrak{w}_i, \mathfrak{w}_j)$.

(III) For all $\mathfrak{w}_0, \mathfrak{w}_1, \mathfrak{w}_2 \in \Sigma^*$, $\mathit{clen}(\mathfrak{w}_0, \mathfrak{w}_1) < \mathit{clen}(\mathfrak{w}_1, \mathfrak{w}_2)$ implies $\mathit{clen}(\mathfrak{w}_0, \mathfrak{w}_1) = \mathit{clen}(\mathfrak{w}_0, \mathfrak{w}_2)$.

Let VAR' be a finite subset of VAR. A string valuation $\mathfrak{s}$ with respect to VAR' is a map $\mathfrak{s} : \mathrm{VAR}' \to \Sigma^*$. A *counter valuation* $\mathfrak{c}$ with respect to VAR' is defined as a map $\mathfrak{c} : \{\mathtt{clen}(\mathtt{x}, \mathtt{x}') : \mathtt{x}, \mathtt{x}' \in \mathrm{VAR}'\} \to \mathbb{N}$, where expressions of the form

$\mathtt{clen}(\mathtt{x},\mathtt{x}')$ are understood as "variables" interpreted on $\mathbb{N}$ (we also adopt a set-theoretical reading: $\mathtt{clen}(\mathtt{x},\mathtt{x}')$ and $\mathtt{clen}(\mathtt{x}',\mathtt{x})$ are considered as identical). In forthcoming Section 4.2, we adopt a similar notation. We say that a counter valuation $\mathfrak{c}$ is *string-compatible* (with respect to VAR') if $\mathfrak{c}$ satisfies the conjunction of the three constraints below in the concrete domain $(\mathbb{N},<,=,(=_n)_{n\in\mathbb{N}})$.

- Formula $\psi_{\mathrm{I}}(\mathrm{VAR}')$ is related to (I): $\bigwedge_{\mathtt{x},\mathtt{x}'\in\mathrm{VAR}'}(\mathtt{clen}(\mathtt{x},\mathtt{x}) \geqslant \mathtt{clen}(\mathtt{x},\mathtt{x}'))$.
- Formula $\psi_{\mathrm{II}}(\mathrm{VAR}')$ is related to (II):

$$\bigwedge_{\mathtt{x}_0,\ldots,\mathtt{x}_k\in\mathrm{VAR}'}((\bigwedge_{i\in[0,k]}(\mathtt{clen}(\mathtt{x}_0,\mathtt{x}_1) < \mathtt{clen}(\mathtt{x}_i,\mathtt{x}_i))) \ \wedge\ \mathtt{clen}(\mathtt{x}_0,\mathtt{x}_1) = \cdots = \mathtt{clen}(\mathtt{x}_0,\mathtt{x}_k)) \ \Rightarrow (\bigvee_{i\neq j\in[1,k]}(\mathtt{clen}(\mathtt{x}_0,\mathtt{x}_1) < \mathtt{clen}(\mathtt{x}_i,\mathtt{x}_j))).$$

- Formula $\psi_{\mathrm{III}}(\mathrm{VAR}')$ is related to (III):

$$\bigwedge_{\mathtt{x},\mathtt{x}',\mathtt{x}''\in\mathrm{VAR}'}(\mathtt{clen}(\mathtt{x},\mathtt{x}') < \mathtt{clen}(\mathtt{x}',\mathtt{x}'')) \Rightarrow (\mathtt{clen}(\mathtt{x},\mathtt{x}') = \mathtt{clen}(\mathtt{x},\mathtt{x}'')).$$

The size of the above conjunction is in $\mathcal{O}(\mathrm{card}(\mathrm{VAR}')^{k+2})$, i.e. polynomial in $\mathrm{card}(\mathrm{VAR}')$, assuming Σ is fixed. If $X \subseteq \mathrm{VAR}'$ and $\mathfrak{c}$ is string-compatible w.r.t. VAR', the restriction of $\mathfrak{c}$ to X is also string-compatible with respect to X.

Let X be a nonempty subset of VAR', $\mathfrak{s}$ be a string valuation and $\mathfrak{c}$ be a counter valuation, both with respect to VAR'. We say that $\mathfrak{c}$ *agrees with* $\mathfrak{s}$ on X (written $\mathfrak{c} \approx_X \mathfrak{s}$) $\stackrel{\text{def}}{\Leftrightarrow}$ $\mathfrak{c}(\mathtt{clen}(\mathtt{x},\mathtt{x}')) = \mathit{clen}(\mathfrak{s}(\mathtt{x}),\mathfrak{s}(\mathtt{x}'))$ for all $\mathtt{x},\mathtt{x}' \in X$ ('clen' is overloaded here, used to define natural number variables and a function on pairs of strings but we hope this does not lead to confusions). So, if $X' \subseteq X$ and $\mathfrak{c} \approx_X \mathfrak{s}$, then $\mathfrak{c} \approx_{X'} \mathfrak{s}$ too. Given a string valuation $\mathfrak{s}$, there is a counter valuation $\mathfrak{c}$ such that $\mathfrak{c} \approx_X \mathfrak{s}$ [17, Lemma 5] and $\mathfrak{c}$ can be defined obviously by: $\mathfrak{c}(\mathtt{clen}(\mathtt{x},\mathtt{x}')) \stackrel{\text{def}}{=} \mathit{clen}(\mathfrak{s}(\mathtt{x}),\mathfrak{s}(\mathtt{x}'))$ for all $\mathtt{x},\mathtt{x}' \in X$.

However, there are counter valuations $\mathfrak{c}$ for which there exists no string valuation $\mathfrak{s}$ with $\mathfrak{c} \approx_X \mathfrak{s}$ (*cf.* Example 2 below). We state below the main property relating string constraints on $\mathcal{D}_\Sigma$ and constraints on $(\mathbb{N},<,=,(=_n)_{n\in\mathbb{N}})$. Namely, for every string-compatible counter valuation $\mathfrak{c}$ for which the restriction to a subset of variables agrees with some string valuation $\mathfrak{s}$, it is possible to extend $\mathfrak{s}$ to all the variables so that $\mathfrak{c}$ agrees with it.

Proposition 2. *[17, Lemma 6] Let $X \neq \varnothing$ and Y be finite and disjoint sets of string variables, $\mathfrak{c}$ be a string-compatible counter valuation with respect to $X \uplus Y$ and $\mathfrak{s} : Y \to \Sigma^*$ be such that $\mathfrak{c} \approx_Y \mathfrak{s}$. Then, there is a string valuation $\mathfrak{s}'$ that is a conservative extension of $\mathfrak{s}$, such that $\mathfrak{c} \approx_{X\uplus Y} \mathfrak{s}'$.*

Example 2. Let $\mathrm{VAR}' = \{\mathtt{x}_1,\mathtt{x}_2,\mathtt{x}_1',\mathtt{x}_2'\}$ and let $\mathfrak{s}(\mathtt{x}_1) = aaba, \mathfrak{s}(\mathtt{x}_2) = aa, \mathfrak{s}(\mathtt{x}_1') = aab$, and $\mathfrak{s}(\mathtt{x}_2') = aaa$. This string valuation satisfies the constraint $\mathtt{x}_2 \prec \mathtt{x}_2' \wedge \mathtt{x}_2 \prec \mathtt{x}_1'$. In Figure 2, we show the counter valuation $\mathfrak{c}$ with respect to VAR' induced by $\mathfrak{s}$; for instance, $\mathfrak{c}(\mathtt{clen}(\mathtt{x}_1,\mathtt{x}_2')) = 2$. Note that $\mathfrak{c}$ satisfies the constraint over $\mathbb{N}$ corresponding to the above string constraint; for instance, $\mathtt{clen}(\mathtt{x}_2,\mathtt{x}_2) < \mathtt{clen}(\mathtt{x}_2',\mathtt{x}_2') \wedge \mathtt{clen}(\mathtt{x}_2,\mathtt{x}_2) = \mathtt{clen}(\mathtt{x}_2,\mathtt{x}_2')$ holds true (the yellow cells). The other three tables show three counter valuations that are not string-compatible:

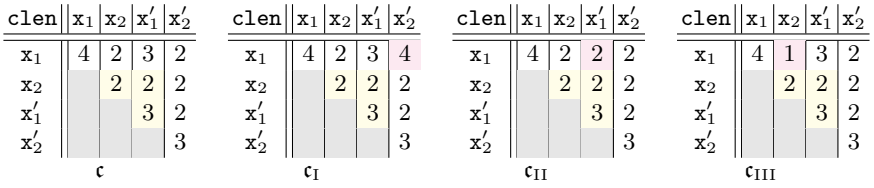

Fig. 2: Counter valuations.

- $< \mathfrak{c}_{\mathrm{I}}(\mathtt{clen}(\mathtt{x}_2', \mathtt{x}_2')) < \mathfrak{c}_{\mathrm{I}}(\mathtt{clen}(\mathtt{x}_1, \mathtt{x}_2'))$ violates constraint $\psi_{\mathrm{I}}(\mathrm{VAR}')$;
- $\mathfrak{c}_{\mathrm{II}}(\mathtt{clen}(\mathtt{x}_2', \mathtt{x}_1')) = \mathfrak{c}_{\mathrm{II}}(\mathtt{clen}(\mathtt{x}_2', \mathtt{x}_1)) = 2$, $\mathfrak{c}_{\mathrm{II}}(\mathtt{clen}(\mathtt{x}_2', \mathtt{x}_2')) > 2$, $\mathfrak{c}_{\mathrm{II}}(\mathtt{clen}(\mathtt{x}_1, \mathtt{x}_1)) > 2$, $\mathfrak{c}_{\mathrm{II}}(\mathtt{clen}(\mathtt{x}_1', \mathtt{x}_1')) > 2$, $\mathfrak{c}_{\mathrm{II}}(\mathtt{clen}(\mathtt{x}_1', \mathtt{x}_2')) = \mathfrak{c}_{II}(\mathtt{clen}(\mathtt{x}_1, \mathtt{x}_1'))$ violate the constraint $\psi_{\mathrm{II}}(\mathrm{VAR}')$, assuming $\mathrm{card}(\Sigma) = 2$.
- $\mathfrak{c}_{\mathrm{III}}(\mathtt{clen}(\mathtt{x}_1, \mathtt{x}_2)) < \mathfrak{c}_{\mathrm{III}}(\mathtt{clen}(\mathtt{x}_2, \mathtt{x}_2'))$ and $\mathfrak{c}_{\mathrm{III}}(\mathtt{clen}(\mathtt{x}_1, \mathtt{x}_2)) < \mathfrak{c}_{\mathrm{III}}(\mathtt{clen}(\mathtt{x}_1, \mathtt{x}_2'))$ violate constraint $\psi_{\mathrm{III}}(\mathrm{VAR}')$.

Even though each of these counter valuations satisfies the constraint corresponding to the string constraint Θ_1, there does not exist any agreeing string valuation. Proposition 2 shows that if a counter valuation is string-compatible, then an agreeing string valuation exists.

4.2 Reducing TCA on strings to TCA on natural numbers

Let $\mathbb{A} = (Q, \mathbf{\Sigma}, d, \beta, Q_{\mathrm{in}}, \delta, F)$ be a TCA on the concrete domain $\mathcal{D}_\Sigma$ for which we wish to check the nonemptiness of $\mathrm{L}(\mathbb{A})$. Below, we define a TCA $\mathbb{A}' = (Q, \mathbf{\Sigma}, d, \beta', Q_{\mathrm{in}}, \delta', F)$ on the concrete domain $(\mathbb{N}, <, =, (=_n)_{n\in\mathbb{N}})$ such that $\mathrm{L}(\mathbb{A})$ is nonempty iff $\mathrm{L}(\mathbb{A}')$ is nonempty. It is worth noting that $\mathbb{A}$ and $\mathbb{A}'$ share the same set of locations, initial locations and the same acceptance condition. Moreover, the finite alphabet $\mathbf{\Sigma}$ and the degree d are identical too. The differences are related to the number of variables β' as well as the definition of the transition relation δ'. Since the two TCA are built over distinct concrete domains ($\mathcal{D}_\Sigma$ versus $\mathbb{N}$), the transition relations necessarily differ. As far as the number of variables is concerned, we lift what is done in Section 4.1 to all the string values occurring in trees accepted by the input TCA $\mathbb{A}$.

- Assume that the constant strings occurring in constraints in δ are $\mathfrak{w}_1, \ldots, \mathfrak{w}_\alpha$ for some $\alpha \geqslant 0$. We write $\mathrm{VAR}_{\mathbb{A}'}$ to denote the expressions in $\{\mathtt{x}_i \mid i \in [1, \beta+\alpha]\} \cup \{S^{-1}\mathtt{x}_i \mid i \in [1, \beta+\alpha]\}$ (not yet variables in $\mathbb{A}'$). Here, $S^{-1}\mathtt{x}$ refers to a value for the parent node, if any. The variables in $\mathbb{A}'$ are of the form $\mathtt{clen}(\mathtt{t}_1, \mathtt{t}_2)$ where $\mathtt{t}_1, \mathtt{t}_2 \in \mathrm{VAR}_{\mathbb{A}'}$ (ad-hoc notation). Consequently, $\beta' = 4(\beta+\alpha)^2$ (polynomial in the size of $\mathbb{A}$). Each string $\mathfrak{w}_i$ from $\mathbb{A}$ is implicitly associated to an expression $\mathtt{x}_{\beta+i}$ in $\mathrm{VAR}_{\mathbb{A}'}$. The variables $\mathtt{clen}(\mathtt{t}_1, \mathtt{t}_2)$'s with $\{\mathtt{t}_1, \mathtt{t}_2\} \cap \{\mathtt{x}_{\beta+i}\} \neq \varnothing$ are intended to specify the length of the longest common prefix between $\mathfrak{w}_i$ and another value.
- The definition of δ' reflects that string-compatible counter valuations satisfy (I)-(III) above, as well as the way we manage the values between the parent

node and its children nodes. Given $(q, \mathtt{a}, (\Theta_0, q_0), \ldots, (\Theta_{d-1}, q_{d-1}))$ in δ, there is a corresponding transition $(q, \mathtt{a}, (\Theta'_0, q_0), \ldots, (\Theta'_{d-1}, q_{d-1}))$ in δ' (leading to $\mathrm{card}(\delta') = \mathrm{card}(\delta)$ by construction). What remains to be done is to define each Θ'_ℓ from Θ_ℓ. Θ'_ℓ is a conjunction made of three conjuncts:

- The 1st conjunct (independent of Θ_ℓ) stating that the counter valuations are string-compatible is equal to $\psi_{\mathrm{I}}(\mathrm{VAR}_{\mathbb{A}'}) \wedge \psi_{\mathrm{II}}(\mathrm{VAR}_{\mathbb{A}'}) \wedge \psi_{\mathrm{III}}(\mathrm{VAR}_{\mathbb{A}'})$, see Section 4.1. Its size is in $\mathcal{O}(\beta+\alpha)^{k+2})$ with $\mathrm{card}(\Sigma) = k$.
- The 2nd conjunct has a double purpose: to define constraints between a node and its parent node (if any), and to guarantee that the expressions of the form $S^{-1}\mathtt{x}_{\beta+i}$ and $\mathtt{x}_{\beta+i}$ can be interpreted as the string $\mathfrak{w}_i$. Here is the 2nd conjunct (also independent of Θ_ℓ):

$$\Big(\bigwedge_{i,j \in [1,\beta+\alpha]} \mathtt{clen}(S^{-1}\mathtt{x}_i, S^{-1}\mathtt{x}_j)' = \mathtt{clen}(\mathtt{x}_i, \mathtt{x}_j) \Big) \wedge$$

$$\Big(\bigwedge_{i,j \in [1,\alpha]} \mathtt{clen}(\mathtt{x}_{\beta+i}, \mathtt{x}_{\beta+j}) = \mathit{clen}(\mathfrak{w}_i, \mathfrak{w}_j) \Big).$$

  Observe that in the equalities in the second conjunct above, the left-hand side is a variable in $\mathbb{A}'$ (using our ad-hoc notation) whereas the right-hand side is a value in $\mathbb{N}$ (clearly bounded by the length of the longest string in $\mathbb{A}$). The size of this conjunct is in $\mathcal{O}((\beta+\alpha)^2 + \alpha^2 \times \max|\mathfrak{w}_i|)$.
- The 3rd conjunct is equal to $\mathfrak{t}(\Theta_\ell)$ where $\mathfrak{t}$ is a translation map that is homomorphic for Boolean connectives. The translation of the atomic constraints is defined in Figure 3, and it takes into account how the values are constrained between a parent node and its child node. For instance, if in some accepted tree $\mathbf{t}$ of $\mathbb{A}$, $\mathbf{t}(\mathbf{n}) = (\mathtt{a}, \vec{z})$ and $\mathbf{t}(\mathbf{n} \cdot \ell) = (\mathtt{a}_\ell, \vec{z}_\ell)$, the variable $\mathtt{clen}(\mathtt{x}_i, S^{-1}\mathtt{x}_j)'$ occuring in Θ'_ℓ refers to the length of the longest common prefix between the value of $\mathtt{x}_i$ at $\mathbf{n} \cdot \ell$ (i.e. $\vec{z}_\ell(i)$) and the value of $\mathtt{x}_j$ at the parent node of $\mathbf{n} \cdot \ell$ (i.e. $\vec{z}(j)$). The size of the 3rd conjunct $\mathfrak{t}(\Theta_\ell)$ is linear in the size of Θ_ℓ.

Example 3. Consider $\mathbb{A}$ from Example 1. The automaton $\mathbb{A}'$ obtained from the above construction is of the form $(\{q, q_\mathtt{a}, q_\mathtt{b}\}, \{\mathtt{a}, \mathtt{b}\}, 2, 16, \{q\}, \delta', \{q_\mathtt{a}, q_\mathtt{b}\})$, where δ' contains the transitions $(q, \mathtt{a}, (\Theta'_0, q_\mathtt{b}), (\Theta'_1, q_\mathtt{a}))$, $(q_\mathtt{a}, \mathtt{a}, (\Theta', q_\mathtt{a}), (\Theta', q_\mathtt{a}))$, and $(q_\mathtt{b}, \mathtt{b}, (\Theta'_0, q_\mathtt{b}), (\Theta'_0, q_\mathtt{b}))$, with Θ'_0, Θ'_1, and Θ' obtained from the corresponding string constraints in $\mathbb{A}$ as described above. Θ'_1 consists (amongst others) of the constraints $\mathtt{clen}(S^{-1}\mathtt{x}_2, S^{-1}\mathtt{x}_2)' = \mathtt{clen}(S^{-1}\mathtt{x}_2, \mathtt{x}_1)' \wedge \mathtt{clen}(S^{-1}\mathtt{x}_2, S^{-1}\mathtt{x}_2)' < \mathtt{clen}(\mathtt{x}_1, \mathtt{x}_1)'$ (the translation of the string constraint $\mathtt{x}_2 \prec \mathtt{x}'_1$). By the 2nd conjunct in the definition of δ', we have $\mathtt{clen}(\mathtt{x}_2, \mathtt{x}_2) = \mathtt{clen}(S^{-1}\mathtt{x}_2, S^{-1}\mathtt{x}_2)'$.

The correctness of the construction of $\mathbb{A}'$ is best illustrated by the statement below, which can be viewed as an automata-based counterpart of [17, Lemma 10] and requires a lengthy proof. It relies on Proposition 2 when new string values need to be considered.

Lemma 1. $\mathrm{L}(\mathbb{A}) \neq \varnothing$ *iff* $\mathrm{L}(\mathbb{A}') \neq \varnothing$.

Atomic θ	Translation $\mathfrak{t}(\theta)$
$\mathtt{x}_i \prec \mathtt{x}_j$	$\mathtt{clen}(\mathtt{x}_i,\mathtt{x}_i) = \mathtt{clen}(\mathtt{x}_i,\mathtt{x}_j) \wedge \mathtt{clen}(\mathtt{x}_i,\mathtt{x}_i) < \mathtt{clen}(\mathtt{x}_j,\mathtt{x}_j)$
$\mathtt{x}_i = \mathtt{x}_j$	$\mathtt{clen}(\mathtt{x}_i,\mathtt{x}_i) = \mathtt{clen}(\mathtt{x}_i,\mathtt{x}_j) = \mathtt{clen}(\mathtt{x}_j,\mathtt{x}_j)$
$\mathtt{x}_i = \mathfrak{w}_j$	$\mathtt{clen}(\mathtt{x}_i,\mathtt{x}_i) = \mathtt{clen}(\mathtt{x}_i,\mathtt{x}_{\beta+j}) = \mathtt{clen}(\mathtt{x}_{\beta+j},\mathtt{x}_{\beta+j})$
$\mathtt{x}'_i \prec \mathtt{x}_j$	$\mathtt{clen}(\mathtt{x}_i,\mathtt{x}_i)' = \mathtt{clen}(\mathtt{x}_i, S^{-1}\mathtt{x}_j)' \wedge$ $\mathtt{clen}(\mathtt{x}_i,\mathtt{x}_i)' < \mathtt{clen}(S^{-1}\mathtt{x}_j, S^{-1}\mathtt{x}_j)'$
$\mathtt{x}'_i = \mathtt{x}_j$	$\mathtt{clen}(\mathtt{x}_i,\mathtt{x}_i)' = \mathtt{clen}(\mathtt{x}_i, S^{-1}\mathtt{x}_j)' = \mathtt{clen}(S^{-1}\mathtt{x}_j, S^{-1}\mathtt{x}_j)'$
$\mathtt{x}'_i = \mathfrak{w}_j$	$\mathtt{clen}(\mathtt{x}_i,\mathtt{x}_i)' = \mathtt{clen}(\mathtt{x}_i,\mathtt{x}_{\beta+j})' = \mathtt{clen}(\mathtt{x}_{\beta+j},\mathtt{x}_{\beta+j})'$
$\mathtt{x}_i \prec \mathtt{x}'_j$	$\mathtt{clen}(S^{-1}\mathtt{x}_i, S^{-1}\mathtt{x}_i)' = \mathtt{clen}(S^{-1}\mathtt{x}_i,\mathtt{x}_j)' \wedge$ $\mathtt{clen}(S^{-1}\mathtt{x}_i, S^{-1}\mathtt{x}_i)' < \mathtt{clen}(\mathtt{x}_j,\mathtt{x}_j)'$
$\mathtt{x}'_i \prec \mathtt{x}'_j$	$\mathtt{clen}(\mathtt{x}_i,\mathtt{x}_i)' = \mathtt{clen}(\mathtt{x}_i,\mathtt{x}_j)' \wedge \mathtt{clen}(\mathtt{x}_i,\mathtt{x}_i)' < \mathtt{clen}(\mathtt{x}_j,\mathtt{x}_j)'$
$\mathtt{x}'_i = \mathtt{x}'_j$	$\mathtt{clen}(\mathtt{x}_i,\mathtt{x}_i)' = \mathtt{clen}(\mathtt{x}_i,\mathtt{x}_j)' = \mathtt{clen}(\mathtt{x}_j,\mathtt{x}_j)'$

Fig. 3: Translation of atomic constraints.

We are ready to present our main result about the complexity of NE(TCA($\mathcal{D}_\Sigma$)).

Theorem 1. *For every finite alphabet Σ,* NE(*TCA*($\mathcal{D}_\Sigma$)) *is* ExpTime*-complete.*

ExpTime-hardness of NE(TCA($\mathcal{D}_\Sigma$)) is due to ExpTime-hardness of the problem NE(TCA($\mathbb{N}$)) established in [19, Section 4.1]. To prove ExpTime-membership, given $\mathbb{A}'$ (on the concrete domain $\mathbb{N}$) built from $\mathbb{A}$ (on the concrete domain $\mathcal{D}_\Sigma$), we know from [19, Lemma 10] that, the nonemptiness of L($\mathbb{A}'$) can be solved in time

$$\mathcal{O}\Big(R_1\big(\mathrm{card}(Q) \times \mathrm{card}(\delta') \times \mathtt{MCS}(\mathbb{A}') \times \mathrm{card}(\mathbf{\Sigma}) \times R_2(\beta')\big)^{R_2(\beta')\times R_3(d)}\Big)$$

for some polynomials R_1, R_2 and R_3, where $\mathtt{MCS}(\mathbb{A}')$ denotes the maximal size of a constraint occurring in $\mathbb{A}'$. Note that the result is stated for the concrete domain $\mathbb{Z}$ in [19, Lemma 10], but it applies to $\mathbb{N}$ too (there is a simple way to enforce $\mathtt{x}_i \geqslant 0$ everywhere). We adopt a similar notation for $\mathbb{A}$ and from the above developments, $\mathtt{MCS}(\mathbb{A}')$ is in $\mathcal{O}((\beta + \mathtt{MCS}(\mathbb{A}) \times \mathit{card}(\delta) \times d)^{k+3})$ as α can be shown to be bounded above by $\mathtt{MCS}(\mathbb{A}) \times \mathrm{card}(\delta) \times d$. Since $\mathrm{card}(\delta') = \mathrm{card}(\delta)$, $\beta' = 4(\beta+\alpha)^2$, nonemptiness of L($\mathbb{A}$) can be solved in time $\mathcal{O}\Big(R_1\big(\mathrm{card}(Q) \times \mathrm{card}(\delta) \times (\beta + \mathtt{MCS}(\mathbb{A}) \times \mathit{card}(\delta) \times d)^{k+3} \times \mathrm{card}(\mathbf{\Sigma}) \times R_2(\beta^\dagger)\big)^{R_2(\beta^\dagger)\times R_3(d)}\Big)$ with $\beta^\dagger = 4(\beta + \mathtt{MCS}(\mathbb{A}) \times \mathrm{card}(\delta) \times d)^2$. Hence, NE(TCA($\mathcal{D}_\Sigma$)) is in ExpTime; this holds even if Σ is part of the input.

5 Automata-Based Approach for $\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_{\mathbf{\Sigma}})$

Below, we reduce TSAT($\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma)$) to NE(TCA): given an $\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma)$ concept C_0 and a TBox $\mathcal{T}$, we construct a TCA $\mathbb{A}$ on $\mathcal{D}_\Sigma$ such that $C_0, \mathcal{T}$ is a positive instance of TSAT($\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma)$) iff L($\mathbb{A}$) $\neq \varnothing$. The material below follows the arguments from [19, Section 5.2] but for $\mathcal{D}_\Sigma$. Obviously, the *tree*

interpretation property of $\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma)$ (cf. the proof of Proposition 1) is of use in the reduction. Thanks to Proposition 1, we can assume that all input concepts are in negation normal form and terms are restricted to those in $\mathrm{T}^{\leqslant 1}_{\mathrm{VAR}}$, that is, the role paths are restricted to single role names r and to ε. This simplifies the reduction; for instance, $\forall\varepsilon.[\![\Theta]\!]$ is logically equivalent to $\exists\varepsilon.[\![\Theta]\!]$, and Θ contains solely variables that state constraints only for the current individual. At this point, it is worth noting that atomic concepts of the form $\exists P.[\![\Theta]\!]$ or $\forall P.[\![\Theta]\!]$ can be expressed by constraints in TCA, unlike the automata-based approach used in [18,27,26] that involves abstractions and finite alphabets only.

Since interpretations for $\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma)$ concepts provide a semantics for several role names, we use a standard trick and reserve directions in $[0, d-1]$ for each role name r occurring in the instance of $\mathrm{TSAT}(\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma))$. This is needed because in the trees $\mathbf{t} : [0, d-1]^* \to \mathbf{\Sigma} \times (\Sigma^*)^\beta$, the (implicit) edges are not labelled. Another way to proceed would be to add a role name to each location of the TCA in order to remember how the node in the tree $[0, d-1]^*$ is accessed to, which is a technique used in [3, Section 3.2]. We also have to handle the determinism of the binary relations $r^{\mathcal{I}}$ with $r \in \mathsf{N}_{\mathbf{F}}$.

So let C_0 be an $\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma)$ concept and let $\mathcal{T}$ be $\{C_1 \sqsubseteq D_1, \ldots, C_\ell \sqsubseteq D_\ell\}$, with the above-mentioned syntactic restrictions. Given $X \subseteq \mathrm{sub}(C_0, \mathcal{T})$, we say that X is *propositionally $\mathcal{T}$-consistent* iff the conditions below hold.

- There is no concept name A such that $\{A, \neg A\} \subseteq X$.
- X does not contain $\bot$ and if $\top \in \mathrm{sub}(C_0, \mathcal{T})$, then $\top \in X$.
- If $E_1 \sqcup E_2 \in X$, then $\{E_1, E_2\} \cap X \neq \varnothing$. if $E_1 \sqcap E_2 \in X$, then $\{E_1, E_2\} \subseteq X$.
- For all $k \in [1, \ell]$, if $C_k \in X$, then $D_k \in X$.

Propositionally $\mathcal{T}$-consistent sets correspond to Hintikka sets from [3, Section 3.2] and their introduction is common for developing an automata-based approach for (description) logics. There is no clause for the concept constructor negation because the concepts are in simple form and negation occurs only in front of concept names or within constraints Θ.

Given $r \in \mathsf{N}_{\mathbf{R}}$, we define $\mathrm{sub}_{\exists r}(C_0, \mathcal{T}) \stackrel{\text{def}}{=} \{\exists r.D \mid \exists r.D \in \mathrm{sub}(C_0, \mathcal{T})\}$ and $\mathrm{sub}_{\exists^{cst} r}(C_0, \mathcal{T}) \stackrel{\text{def}}{=} \{\exists r.[\![\Theta]\!] \mid \exists r.[\![\Theta]\!] \in \mathrm{sub}(C_0, \mathcal{T})\}$. The superscript '$cst$' in $\exists^{cst}$ is intended to remind us that the respective sets are made of atomic concepts involving constraints in the string domain (a.k.a. predicate restrictions). We similarly define $\mathrm{sub}_{\forall r}(C_0, \mathcal{T})$ and $\mathrm{sub}_{\forall^{cst} r}(C_0, \mathcal{T})$. We further define $\mathsf{N}_{\mathbf{F}}(C_0, \mathcal{T}) \stackrel{\text{def}}{=} \{r \in \mathsf{N}_{\mathbf{F}} \mid \mathrm{sub}_{\exists r}(C_0, \mathcal{T}) \cup \mathrm{sub}_{\exists^{cst} r}(C_0, \mathcal{T}) \neq \varnothing\}$ and $\exists_{\overline{\mathbf{F}}}(C_0, \mathcal{T}) \stackrel{\text{def}}{=} \{\exists r.[\![\Theta]\!] \in \mathrm{sub}(C_0, \mathcal{T}) \mid r \notin \mathsf{N}_{\mathbf{F}}(C_0, \mathcal{T})\} \cup \{\exists r.D \in \mathrm{sub}(C_0, \mathcal{T}) \mid r \notin \mathsf{N}_{\mathbf{F}}(C_0, \mathcal{T})\}$. So $\mathsf{N}_{\mathbf{F}}(C_0, \mathcal{T})$ contains functional role names r related to predicate or existential restrictions from $\mathrm{sub}(C_0, \mathcal{T})$ involving r, $\exists_{\overline{\mathbf{F}}}(C_0, \mathcal{T})$ contains predicate or existential restrictions from $\mathrm{sub}(C_0, \mathcal{T})$ involving non-functional role names. This difference of treatment is handy to define the value d below though we need to distinguish in several definitions functional role names from the other ones.

Set $d = \mathrm{card}(\mathsf{N}_{\mathbf{F}}(C_0, \mathcal{T})) + \mathrm{card}(\exists_{\overline{\mathbf{F}}}(C_0, \mathcal{T}))$ and ι be a bijection $\iota : \big(\mathsf{N}_{\mathbf{F}}(C_0, \mathcal{T}) \cup \exists_{\overline{\mathbf{F}}}(C_0, \mathcal{T})\big) \to [1, d]$. We write $r \rhd j$ whenever $\iota^{-1}(j) = r$ or $\iota^{-1}(j)$ is of the form either $\exists r.D$ or $\exists r.[\![\Theta]\!]$ (direction j contributes to witnesses for r).

We build a TCA $\mathbb{A} = (Q, \mathbf{\Sigma}, d+1, \beta, Q_{\text{in}}, \delta, F)$ such that $C_0, \mathcal{T}$ is a positive instance iff $\text{L}(\mathbb{A}) \neq \varnothing$. The automaton $\mathbb{A}$ accepts infinite trees of the form $\mathbf{t} : [0,d]^* \to \mathbf{\Sigma} \times (\Sigma^*)^\beta$ where $\mathbf{\Sigma} = \mathcal{P}(\{A_1, \ldots, A_M\})$, $\{A_1, \ldots, A_M\}$ being the set of concept names occurring in $C_0, \mathcal{T}$. Let us define $\mathbb{A}$ formally.

- Q is the set of propositionally $\mathcal{T}$-consistent subsets of $\text{sub}(C_0, \mathcal{T})$ plus the distinguished "dead-end" location $\bot$ (and never $D \in \bot$, for all concepts D). $\bot$ is useful as seriality is not required for the interpretation of role names.
- $Q_{\text{in}} \stackrel{\text{def}}{=} \{Y \in Q \mid C_0 \in Y\}$, $F \stackrel{\text{def}}{=} Q$ (all the locations are accepting similarly to looping automata, see e.g. [6, Section 3.2]).
- The transition relation δ is made of tuples $(Y, X, (\Theta_0, Y_0), \ldots, (\Theta_d, Y_d))$ s.t.:
  1. For all $A \in Y$, we have $A \in X$ and for all $\neg A \in Y$, we have $A \notin X$.
  2. If $Y = \bot$, then $Y_0 = \cdots = Y_d = \bot$.
  3. For all $j \in [1,d]$ such that $Y_j = \bot$, (a) if $\iota^{-1}(j) = r$ for some $r \in \mathsf{N_F}$, then Y has no concepts of the form either $\exists r.D$ or $\exists r.[\![\Theta]\!]$ and (b) if $\iota^{-1}(j) \notin \mathsf{N_F}$ then $\iota^{-1}(j) \notin Y$.
  4. For all $\exists r.D \in Y$, we have either ($r \in \mathsf{N_F}$ and $D \in Y_{\iota(r)}$) or ($r \notin \mathsf{N_F}$ and $D \in Y_{\iota(\exists r.D)}$). The direction to satisfy $\exists r.D$ is either $\iota(r)$ or $\iota(\exists r.D)$.
  5. For all $\forall r.D \in Y$ and $j \in [1,d]$ such that $Y_j \neq \bot$ and $r \rhd j$, we have $D \in Y_j$. In this case, the direction j is reserved for the role name r and for obligations related to the satisfaction of $\forall r.D$. The satisfaction of $\forall r.D$ implies the satisfaction of D for all the $r^{\mathcal{I}}$-successors, if any.
  6. For all $j \in [0,d]$, the constraint Θ_j is defined as follows.
     (a) If $Y = \bot$, then $\Theta_j \stackrel{\text{def}}{=} \top$.
     (b) Otherwise, if $j = 0$ or $Y_j = \bot$, then $\Theta_j \stackrel{\text{def}}{=} (\bigwedge_{\exists \varepsilon.[\![\Theta']\!], \forall \varepsilon.[\![\Theta']\!] \in Y} \Theta')$. This conjunction needs actually to be satisfied whenever $Y_j \neq \bot$.
     (c) Otherwise, if $\big((\exists r.[\![\Theta]\!] \notin Y$ and $\iota(\exists r.[\![\Theta]\!]) = j)$ or $\iota(\exists r.D) = j$ for some $\exists r.[\![\Theta]\!], \exists r.D \in \exists_{\overline{\mathbf{F}}}(C_0, \mathcal{T})\big)$ and $Y_j \neq \bot$ (necessarily $r \notin \mathsf{N_F}$), then $\Theta_j \stackrel{\text{def}}{=} (\bigwedge_{\exists \varepsilon.[\![\Theta']\!], \forall \varepsilon.[\![\Theta']\!] \in Y} \Theta') \wedge (\bigwedge_{\forall r.[\![\Theta']\!] \in Y} \Theta')$.
     (d) Otherwise, if there is $\exists r.[\![\Theta]\!] \in Y$ s.t. $\iota(\exists r.[\![\Theta]\!]) = j$ (necessarily $r \notin \mathsf{N_F}$), $\Theta_j \stackrel{\text{def}}{=} (\bigwedge_{\exists \varepsilon.[\![\Theta']\!], \forall \varepsilon.[\![\Theta']\!] \in Y} \Theta') \wedge (\bigwedge_{\forall r.[\![\Theta']\!] \in Y} \Theta') \wedge \Theta$. By (3.) above, $\exists r.[\![\Theta]\!] \in Y$, $\iota(\exists r.[\![\Theta]\!]) = j$ and $r \notin \mathsf{N_F}$ imply $Y_j \neq \bot$. So, the direction j is reserved for the role name r and for obligations related to the satisfaction of $\exists r.[\![\Theta]\!]$. This case occurs if there are obligations to satisfy $\exists r.[\![\Theta]\!]$ whereas the case 6(b) gives more freedom because the satisfaction of $\exists r.[\![\Theta]\!]$ is not imposed from Y.
     (e) Otherwise, i.e. $r = \iota^{-1}(j) \in \mathsf{N_F}$ and $Y_j \neq \bot$,

$$\Theta_j \stackrel{\text{def}}{=} \Big(\bigwedge_{\exists \varepsilon.[\![\Theta']\!], \forall \varepsilon.[\![\Theta']\!] \in Y} \Theta'\Big) \wedge \Big(\bigwedge_{\forall r.[\![\Theta']\!], \exists r.[\![\Theta']\!] \in Y} \Theta'\Big).$$

     Unlike the previous cases, if $r \in \mathsf{N_F}$, then $\forall r.[\![\Theta']\!]$ and $\exists r.[\![\Theta']\!]$ are logically equivalent, assuming that there is one $r^{\mathcal{I}}$-successor.

We can show that our construction is correct.

Lemma 2. *$C_0, \mathcal{T}$ is a positive instance of* $\text{TSAT}(\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma))$ *iff* $\text{L}(\mathbb{A}) \neq \varnothing$.

Despite the involved construction of $\mathbb{A}$ due to the expressiveness of the logic $\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma)$, the proof follows a standard pattern. If $C_0, \mathcal{T}$ is a positive instance, then we can extract a tree interpretation that can be associated to a tree accepted by $\mathbb{A}$. Conversely, any tree accepted by $\mathbb{A}$ can be turned into a tree interpretation witnessing the satisfaction of $C_0, \mathcal{T}$. The result below is the main technical result in this paper, whose proof combines the previous key lemmas.

Theorem 2. *For every finite Σ,* $\mathrm{TSAT}(\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma))$ *is* ExpTime*-complete.*

Due to our complexity analysis, the ExpTime-membership is preserved if the alphabet Σ is part of the input (and not a parameter as in $\mathrm{TSAT}(\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma))$).

Concluding remarks. We have shown that the nonemptiness problem for tree constraint automata on $\mathcal{D}_\Sigma$ is ExpTime-complete (Theorem 1) and the concept satisfiability problem w.r.t. general TBoxes for $\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma)$ is ExpTime-complete too (Theorem 2). The suite of key steps is schematised below.

$$\mathrm{TSAT}(\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma)) \xrightarrow{\text{Prop. 1}} \underset{\text{in simple form}}{\mathrm{TSAT}(\mathcal{ALCF}^{\mathcal{P}}(\mathcal{D}_\Sigma))} \xrightarrow{\text{Lemma 2}} \mathrm{NE}(\mathrm{TCA}(\mathcal{D}_\Sigma)) \xrightarrow{\text{Lemma 1}} \mathrm{NE}(\mathrm{TCA}(\mathbb{N}))$$

These are only first steps to handle more concrete domains based on strings and on richer description logics. Typically, though we believe we could generalise the developments herein to mix numerical constraints and prefix constraints or to admit an infinite alphabet (based on developments in [17]), it is unclear how to handle $\mathcal{D}_\Sigma$ within logics from [29] (see also [27, Section 4.1]). Similarly, it is open how to handle the string domain $\mathcal{D}_\Sigma$ with regularity constraints, to name another possibility for future work.

References

1. Abdulla, P., Atig, M., Chen, Y., Holík, L., Rezine, A., Rümmer, P., Stenman, J.: String constraints for verification. In: CAV'14. LNCS, vol. 8559, pp. 150–166. Springer (2014)
2. Abdulla, P., Atig, M., Chen, Y., Diep, B., Dolby, J., Janku, P., Lin, H., Holik, L., Wu, W.: Efficient handling of string-number conversion. In: PLDI'20. pp. 943–957. ACM (2020)
3. Baader, F.: Description Logics. In: Reasoning Web. Semantic Technologies for Information Systems, 5th International Summer School 2009, Tutorial Lectures. LNCS, vol. 5689, pp. 1–39. Springer (2009)
4. Baader, F., Calvanese, D., Guinness, D.M., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
5. Baader, F., Hanschke, P.: A scheme for integrating concrete domains into concept languages. In: IJCAI'91. pp. 452–457 (1991)
6. Baader, F., Hladik, J., Lutz, C., Wolter, F.: From Tableaux to Automata for Description Logics. Fundamenta Informaticae **57**(2–4), 247–279 (2003)
7. Baader, F., Horrocks, I., Lutz, C., Sattler, U.: An Introduction to Description Logic. Cambridge University Press (2017)

8. Baader, F., Rydval, J.: An Algebraic View on p-Admissible Concrete Domains for Lightweight Description Logics. In: JELIA'21. LNCS, vol. 12678, pp. 194–209. Springer (2021)
9. Baader, F., Rydval, J.: Using model theory to find decidable and tractable description logics with concrete domains. JAR **66**(3), 357–407 (2022)
10. Baader, F., Sattler, U.: Description logics with concrete domains and aggregation. In: ECAI'98. pp. 336–340. John Wiley and Sons (1998)
11. Balbiani, P., Condotta, J.: Computational complexity of propositional linear temporal logics based on qualitative spatial or temporal reasoning. In: FroCoS'02. LNAI, vol. 2309, pp. 162–173. Springer (2002)
12. Bednarczyk, B., Fiuk, O.: Presburger Büchi tree automata with applications to logics with expressive counting. In: WoLLIC'22. LNCS, vol. 13468, pp. 295–308. Springer (2022)
13. Carapelle, C., Feng, S., Kartzow, A., Lohrey, M.: Satisfiability of ECTL* with local tree constraints. Theory Computing Systems **61**(2), 689–720 (2017)
14. Carapelle, C., Kartzow, A., Lohrey, M.: Satisfiability of ECTL* with constraints. Journal of Computer and System Sciences **82**(5), 826–855 (2016)
15. Carapelle, C., Turhan, A.: Description Logics Reasoning w.r.t. General TBoxes is Decidable for Concrete Domains with the EHD-property. In: ECAI'16. vol. 285, pp. 1440–1448. IOS Press (2016)
16. Čerāns, K.: Deciding properties of integral relational automata. In: ICALP'94. LNCS, vol. 820, pp. 35–46. Springer (1994)
17. Demri, S., Deters, M.: Temporal logics on strings with prefix relation. JLC **26**, 989–1017 (2016)
18. Demri, S., D'Souza, D.: An automata-theoretic approach to constraint LTL. I & C **205**(3), 380–415 (2007)
19. Demri, S., Quaas, K.: Constraint automata on infinite data trees: From CTL(Z)/CTL*(Z) to decision procedures. CoRR, abs/2302.05327 (2023)
20. Diekert, V., Gutierrez, C., Hagenah, C.: The existential theory of equations with rational constraints in free groups is PSPACE-complete. I & C **202** (2005)
21. Gascon, R.: An automata-based approach for CTL* with constraints. Electronic Notes in Theoretical Computer Science **239**, 193–211 (2009)
22. Geatti, L., Gianola, A., Gigante, N.: Linear temporal logic modulo theories over finite traces. In: IJCAI'22. pp. 2641–2647. ijcai.org (2022)
23. Haarslev, V., Möller, R.: Description Logic Systems with Concrete Domains: Applications for the Semantic Web. In: KRDB'03. CEUR Workshop Proceedings, vol. 79. CEUR-WS.org (2003)
24. Hustadt, U., Motik, B., Sattler, U.: Reasoning in Description Logics with a Concrete Domain in the Framework of Resolution. In: ECAI'04. pp. 353–357. IOS Press (2004)
25. Kartzow, A., Weidner, T.: Model checking constraint LTL over trees. CoRR, abs/1504.06105 (2015)
26. Labai, N.: Automata-based reasoning for decidable logics with data values. Ph.D. thesis, TU Wien (May 2021)
27. Labai, N., Ortiz, M., Simkus, M.: An Exptime Upper Bound for $\mathcal{ALC}$ with integers. In: KR'20. pp. 425–436. Morgan Kaufman (2020)
28. Liang, T., Reynolds, A., Tinelli, C., Barrett, C., Deters, M.: A DPLL(T) theory solver for a theory of strings and regular expressions. In: CAV'14. LNCS, vol. 8559, pp. 646–662 (2014)
29. Lutz, C.: NEXPTIME-complete description logics with concrete domains. In: IJCAR'01. LNCS, vol. 2083, pp. 46–60. Springer (2001)

30. Lutz, C.: The Complexity of Description Logics with Concrete Domains. Ph.D. thesis, RWTH, Aachen (2002)
31. Lutz, C.: Description logics with concrete domains—a survey. In: Advances in Modal Logics Volume 4. pp. 265–296. King's College Publications (2003)
32. Lutz, C.: NEXPTIME-complete description logics with concrete domains. ACM ToCL **5**(4), 669–705 (2004)
33. Lutz, C., Milićić, M.: A Tableau Algorithm for Description Logics with Concrete Domains and General Tboxes. JAR **38**(1-3), 227–259 (2007)
34. Makanin, G.: The problem of solvability of equations in a free semigroup (english translation). Mathematics of the USSR-Sbornik **32**(2), 129–198 (feb 1977)
35. Peteler, D., Quaas, K.: Deciding Emptiness for Constraint Automata on Strings with the Prefix and Suffix Order. In: MFCS'22. LIPIcs, vol. 241, pp. 76:1–76:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022)
36. Plandowski, W.: Satisfiability of word equations with constants is in PSPACE. Journal of the Association for Computing Machinery **51**(3), 483–496 (2004)
37. Quine, W.: Concatenation as a basis for arithmetic. The Journal of Symbolic Logic **11**(4), 105–114 (1946)
38. Revesz, P.: Introduction to Constraint Databases. Springer, New York (2002)
39. Rydval, J.: Using Model Theory to Find Decidable and Tractable Description Logics with Concrete Domains. Ph.D. thesis, Dresden University (2022)
40. Segoufin, L., Toruńczyk, S.: Automata based verification over linearly ordered data domains. In: STACS'11. pp. 81–92 (2011)
41. Seidl, H., Schwentick, T., Muscholl, A.: Counting in trees. In: Logic and Automata: History and Perspectives. Texts in Logic and Games, vol. 2, pp. 575–612. Amsterdam University Press (2008)
42. Vardi, M., Wolper, P.: Automata-theoretic techniques for modal logics of programs. Journal of Computer and System Sciences **32**, 183–221 (1986)
43. Weidner, T.: Probabilistic Logic, Probabilistic Regular Expressions, and Constraint Temporal Logic. Ph.D. thesis, University of Leipzig (2016)
44. Wolper, P.: On the relation of programs and computations to models of temporal logic. In: Temporal Logic in Specifications. LNCS, vol. 398, pp. 75–123. Springer (1987)