Constraint Automata on Infinite Data Trees: from CTL( )/ CTL*( ) to Decision Procedures

Stéphane Demri, Karin Quaas

Constraint Automata on Infinite Data Trees: From CTL($\mathbb{Z}$)/CTL*($\mathbb{Z}$) To Decision Procedures

Stéphane Demri
Université Paris-Saclay, LMF, CNRS, ENS Paris-Saclay, France

Karin Quaas[1]
Universität Leipzig, Fakultät für Mathematik und Informatik, Germany

Abstract

We introduce the class of tree constraint automata with data values in $\mathbb{Z}$ (equipped with the less than relation and equality predicates to constants), and we show that the nonemptiness problem is ExpTime-complete. Using an automata-based approach, we establish that the satisfiability problem for CTL($\mathbb{Z}$) (CTL with constraints in $\mathbb{Z}$) is ExpTime-complete, and the satisfiability problem for CTL*($\mathbb{Z}$) is 2ExpTime-complete (only decidability was known so far). By-product results with other concrete domains and other logics, are also briefly discussed.

2012 ACM Subject Classification Theory of computation → Logic and verification

Keywords and phrases Constraints, Constraint Automata, Temporal Logics, Infinite Data Trees

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2023.29

1 Introduction

In this paper, we study the satisfiability problem for the branching-time temporal logics CTL($\mathbb{Z}$) and CTL*($\mathbb{Z}$), extending the classical temporal logics CTL and CTL* in that atomic formulae express constraints about the relational structure $(\mathbb{Z}, <, =, (=_\mathfrak{d})_{\mathfrak{d}\in\mathbb{Z}})$. Formulae in these logics are interpreted over Kripke structures that are annotated with values in $\mathbb{Z}$. A typical CTL*($\mathbb{Z}$) formula is the formula $\mathsf{AGF}(\mathtt{x} < \mathsf{X}\mathtt{x})$ stating that on all paths infinitely often the value of the variable $\mathtt{x}$ at the current position is strictly smaller than the value of $\mathtt{x}$ at the next position. Formalisms defined over relational structures, also known as *concrete domains*, are considered in many works, including works on temporal logics [40, 13, 54, 48, 19, 35], description logics [50, 51, 52, 53, 16, 45, 3], and automata [38, 61, 43, 71, 65, 57]. Combining reasoning in your favourite logic with reasoning in a relevant concrete domain reveals to be essential for numerous applications, for instance for reasoning about ontologies, see e.g. [52, 46], or data-aware systems, see e.g. [28, 34]. A brief survey can be found in [26].

Decidability results for concrete domains handled in [53, 38, 3] exclude the ubiquitous concrete domain $(\mathbb{Z}, <, =, (=_\mathfrak{d})_{\mathfrak{d}\in\mathbb{Z}})$. By contrast, decidability results for logics with concrete domain $\mathbb{Z}$ require dedicated proof techniques, see e.g. [11, 23, 61, 46]. In particular, *fragments* of CTL*($\mathbb{Z}$) are shown decidable in [11] using integral relational automata from [17], and the satisfiability problem for existential and universal CTL* with gap-order constraints (more general than the ones in this paper) can be solved in PSpace [12, Theorem 14]. Another important breakthrough came with the decidability of CTL*($\mathbb{Z}$) [15, Theorem 32] (see also [14]) by designing a reduction to a decidable second-order logic, whose formulae are made of Boolean combinations of formulae from MSO and from WMSO+U [10], where U is the unbounding second-order quantifier, see e.g. [8, 9]. This is all the more remarkable as the decidability result is part of a powerful general approach [15], but no sharp complexity upper bound can be inferred. More recently, the condition $C_\mathbb{Z}$ [23] to approximate the set

[1] Supported by the Deutsche Forschungsgemeinschaft (DFG), project 504343613.

of satisfiable symbolic models of a given LTL($\mathbb{Z}$) formula is extended to the branching case in [46] leading to the ExpTime-easiness of a major reasoning task for the description logic $\mathcal{ALCF}^{\mathcal{P}}(\mathbb{Z}_c)$. However, no elementary complexity upper bounds for the satisfiability problem for CTL($\mathbb{Z}$) nor CTL*($\mathbb{Z}$) were known since their decidability was established in [13, 15].

In this paper, we prove that the satisfiability problem for CTL($\mathbb{Z}$) is ExpTime-complete, and the satisfiability problem for CTL*($\mathbb{Z}$) is 2ExpTime-complete. We pursue the *automata-based approach* for solving decision problems for temporal logics, following seminal works for temporal logics, see e.g. [68, 69, 44]. This popular approach consists of reducing logical problems (satisfiability, model-checking) to automata-based decision problems while taking advantage of existing results and decision procedures from automata theory, see e.g. [67].

It is well-known that decision procedures for CTL* are difficult to design, and the combination with the concrete domain $\mathbb{Z}$ is definitely challenging. Moreover, we aim at proposing a general framework: we do not wish for every new logic with concrete domain to study again and again what is the proper way to define products of automata leading to optimal complexity. That is why our main goal in this work is to investigate a new class of *tree constraint automata*, understood as a target formalism in the pure tradition of the automata-based approach, and easy to reuse. The structures accepted by such tree constraint automata are *infinite* trees in which nodes are labelled by a letter from a finite alphabet and a tuple in $\mathbb{Z}^\beta$ for some $\beta \geqslant 1$ (this excludes the automata designed in [36, 37] dedicated to finite trees where no predicate $<$ is involved). Decision problems for alternating automata over infinite alphabets are often undecidable, see e.g. [56, 47, 25, 41], and therefore we advocate the introduction of *nondeterministic* constraint automata without alternation. Our definition of tree constraint automata naturally extends the definition of constraint automata for words (see e.g. [17, 59, 61, 43, 57]) and as far as we know, the extension to infinite trees in the way done herein has not been considered earlier in the literature.

As a key result, we show that the nonemptiness problem for tree constraint automata over $(\mathbb{Z}, <, =, (=_\mathfrak{d})_{\mathfrak{d}\in\mathbb{Z}})$ is ExpTime-complete. In order to obtain the ExpTime upper bound, we adapt results from [46, 45] (originally expressed in the context of interpretations for description logics) and we take advantage of several automata-based constructions for Rabin/Streett tree automata. As a corollary, we establish that the satisfiability problem for CTL($\mathbb{Z}$) is ExpTime-complete (Theorem 14), which is one of the main results of the paper. As a by-product, it also allows us to conclude that the concept satisfiability problem w.r.t. general TBoxes for $\mathcal{ALCF}^{\mathcal{P}}(\mathbb{Z}_c)$ is in ExpTime, a result known since [46].

Our main contribution is the characterisation of the complexity for CTL*($\mathbb{Z}$) satisfiability, which is an open problem evoked in [15, Section 9] and [46, Section 5] (decidability was established ten years ago in [14]). In Section 6, we show that the satisfiability problem for CTL*($\mathbb{Z}$) is in 2ExpTime by using Rabin tree constraint automata (introduced herein). We have to check that the essential steps for CTL* can be lifted to CTL*($\mathbb{Z}$) to get the optimal upper bound. In general, our contributions stem from the cross-fertilisation of automata-based techniques for temporal logics and reasoning about (infinite) structures made of $\mathbb{Z}$-constraints.

A complete version with all the proofs can be found in [27].

2 Temporal Logics with Numerical Domains

2.1 Concrete Domain $(\mathbb{Z}, <, =, (=_\mathfrak{d})_{\mathfrak{d}\in\mathbb{Z}})$ and Kripke Structures

In the sequel, we consider the concrete domain $(\mathbb{Z}, <, =, (=_\mathfrak{d})_{\mathfrak{d}\in\mathbb{Z}})$ (also written $\mathbb{Z}$), where $=_\mathfrak{d}$ is a unary predicate stating the equality with the constant $\mathfrak{d}$ and, $<$ and $=$ are the usual

relations on $\mathbb{Z}$. Let $\mathrm{VAR} = \{\mathtt{x}, \mathtt{y}, \ldots\}$ be a countably infinite set of variables. A *term* $\mathtt{t}$ *over* VAR is an expression of the form $\mathsf{X}^i\mathtt{x}$, where $\mathtt{x} \in \mathrm{VAR}$ and X^i is a (possibly empty) sequence of i symbols 'X'. A term $\mathsf{X}^i\mathtt{x}$ should be understood as a variable (that needs to be interpreted) but, later on, we will see that the prefix X^i will have a temporal interpretation. We write $\mathrm{T}_{\mathrm{VAR}}$ to denote the set of all terms over VAR. For all $i \in \mathbb{N}$, we write $\mathrm{T}^{\leqslant i}_{\mathrm{VAR}}$ to denote the subset of terms of the form $\mathsf{X}^j\mathtt{x}$, where $j \leqslant i$. For instance, $\mathrm{T}^{\leqslant 0}_{\mathrm{VAR}} = \mathrm{VAR}$. An *atomic constraint* θ *over* $\mathrm{T}_{\mathrm{VAR}}$ is an expression of one of the forms below:

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

where $\mathfrak{d} \in \mathbb{Z}$ and $\mathtt{t}, \mathtt{t}' \in \mathrm{T}_{\mathrm{VAR}}$. A *constraint* Θ is defined as a Boolean combination of atomic constraints. Constraints are interpreted on valuations $\mathfrak{v} : \mathrm{T}_{\mathrm{VAR}} \to \mathbb{Z}$ that assign elements from $\mathbb{Z}$ to the terms in $\mathrm{T}_{\mathrm{VAR}}$, so that $\mathfrak{v}$ *satisfies* θ, written $\mathfrak{v} \models \theta$, if and only if, the interpretation of the terms in θ makes θ true in $\mathbb{Z}$ in the usual way. The Boolean connectives are interpreted as usual. A constraint Θ is *satisfiable* $\overset{\text{def}}{\Leftrightarrow}$ there is a valuation $\mathfrak{v} : \mathrm{T}_{\mathrm{VAR}} \to \mathbb{Z}$ such that $\mathfrak{v} \models \Theta$. Similarly, a constraint Θ_1 *entails* a constraint Θ_2 (written $\Theta_1 \models \Theta_2$) $\overset{\text{def}}{\Leftrightarrow}$ for all valuations $\mathfrak{v}$, we have $\mathfrak{v} \models \Theta_1$ implies $\mathfrak{v} \models \Theta_2$. The satisfiability problem restricted to finite conjunctions of atomic constraints can be solved in PTime (see e.g. [17, Lemma 5.5]) and entailment is in coNP. In the sequel, quite often, the valuations $\mathfrak{v}$ are of the form $\{\mathtt{x}_1, \ldots, \mathtt{x}_\beta\} \to \mathbb{Z}$ when we are only interested in the values for the variables in $\{\mathtt{x}_1, \ldots, \mathtt{x}_\beta\}$.

Kripke structures. In order to define logics with the concrete domain $\mathbb{Z}$, the semantical structures of such logics are enriched with valuations that interpret the variables by elements in $\mathbb{Z}$. A $\mathbb{Z}$*-decorated Kripke structure* (or *Kripke structure* for short) $\mathcal{K}$ is a triple $(\mathcal{W}, \mathcal{R}, \mathfrak{v})$, where $\mathcal{W}$ is a non-empty set of *worlds*, $\mathcal{R} \subseteq \mathcal{W} \times \mathcal{W}$ is the accessibility relation and $\mathfrak{v} : \mathcal{W} \times \mathrm{VAR} \to \mathbb{Z}$ is a valuation function. A Kripke structure $\mathcal{K}$ is *total* whenever for all $w \in \mathcal{W}$, there is $w' \in \mathcal{W}$ such that $(w, w') \in \mathcal{R}$. Given a Kripke structure $\mathcal{K} = (\mathcal{W}, \mathcal{R}, \mathfrak{v})$ and a world $w \in \mathcal{W}$, an *infinite path* π from w is an ω-sequence $w_0, w_1 \ldots w_n, \ldots$ such that $w_0 = w$ and for all $i \in \mathbb{N}$, we have $(w_i, w_{i+1}) \in \mathcal{R}$. Finite paths are defined accordingly.

Labelled trees. Given $D \geqslant 1$, a *labelled tree of degree* D is a map $\mathbb{t} : \mathit{dom}(\mathbb{t}) \to \Sigma$ where Σ is some (potentially infinite) alphabet and $\mathit{dom}(\mathbb{t})$ is an infinite subset of $[0, D-1]^*$ such that $\mathbf{n} \in \mathit{dom}(\mathbb{t})$ and $\mathbf{n} \cdot i \in \mathit{dom}(\mathbb{t})$ for all $0 \leqslant i < j$ whenever $\mathbf{n} \cdot j \in \mathit{dom}(\mathbb{t})$ for some $\mathbf{n} \in [0, D-1]^*$ and $j \in [0, D-1]$. The elements of $\mathit{dom}(\mathbb{t})$ are called *nodes*. The empty word ε is the *root node* of $\mathbb{t}$. For every $\mathbf{n} \in \mathit{dom}(\mathbb{t})$, the elements $\mathbf{n} \cdot i$ ($i \in [0, D-1]$) 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 $\mathbb{t}$ 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 $\mathbb{t}$ and a node $\mathbf{n}$ in $\mathit{dom}(\mathbb{t})$, an infinite *path* in $\mathbb{t}$ 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 $\mathbf{n} \cdot j_1 \ldots j_i \in \mathit{dom}(\mathbb{t})$ for all $i \geqslant 1$.

A *tree Kripke structure* $\mathcal{K}$ is a Kripke structure $(\mathcal{W}, \mathcal{R}, \mathfrak{v})$ such that $(\mathcal{W}, \mathcal{R})$ is a tree (not necessarily a full D-ary tree). Tree Kripke structures $(\mathcal{W}, \mathcal{R}, \mathfrak{v})$ such that $(\mathcal{W}, \mathcal{R})$ is isomorphic to the tree induced by $[0, D-1]^*$ are represented by maps of the form $\mathbb{t} : [0, D-1]^* \to \mathbb{Z}^\beta$. This assumes that we only care about the value of the variables $\mathtt{x}_1, \ldots, \mathtt{x}_\beta$ and $\mathbb{t}(\mathbf{n}) = (\mathfrak{d}_1, \ldots, \mathfrak{d}_\beta)$ encodes that for all $i \in [1, \beta]$, we have $\mathfrak{v}(\mathbf{n}, \mathtt{x}_i) = \mathfrak{d}_i$.

2.2 The Logic CTL*($\mathbb{Z}$)

We introduce the logic CTL*($\mathbb{Z}$) extending the temporal logic CTL* from [29] but with constraints over $\mathbb{Z}$. *State formulae* ϕ and *path formulae* Φ of CTL*($\mathbb{Z}$) are defined below

$$\phi := \neg\phi \mid \phi \wedge \phi \mid \mathsf{E}\Phi \qquad \Phi := \phi \mid \mathtt{t} = \mathfrak{d} \mid \mathtt{t}_1 = \mathtt{t}_2 \mid \mathtt{t}_1 < \mathtt{t}_2 \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathsf{X}\Phi \mid \Phi\mathsf{U}\Phi,$$

where $\mathtt{t}, \mathtt{t}_1, \mathtt{t}_2 \in \mathrm{T}_{\mathrm{VAR}}$. The size of a formula is understood as its number of symbols with integers encoded with a binary representation. We use also the universal path quantifier A

and the standard temporal connectives R and G ($\mathsf{A}\Phi \stackrel{\text{def}}{=} \neg\mathsf{E}\neg\Phi$, $\Phi_1\mathsf{R}\Phi_2 \stackrel{\text{def}}{=} \neg(\neg\Phi_1\mathsf{U}\neg\Phi_2)$, and $\mathsf{G}\Phi \stackrel{\text{def}}{=} \bot\,\mathsf{R}\,\Phi$ with $\bot$ equal to $\mathsf{E}(\mathtt{x} < \mathtt{x})$). No propositional variables occur in CTL*($\mathbb{Z}$) formulae, but it is easy to simulate them with atomic formulae of the form $\mathsf{E}(\mathtt{x} = 0)$. We say that a formula in CTL*($\mathbb{Z}$) is in *simple form* if it is in negation normal form (using A, R and $\vee$ as primitive) and all terms occurring in the formula are from $\mathrm{T}^{\leqslant 1}_{\mathrm{VAR}}$. State formulae are interpreted on worlds from a Kripke structure, whereas path formulae are interpreted on infinite paths. The two satisfaction relations are defined as follows (we omit the clauses for Boolean connectives), where $\mathcal{K} = (\mathcal{W}, \mathcal{R}, \mathfrak{v})$ is a total Kripke structure, and $w \in \mathcal{W}$.

- $\mathcal{K}, w \models \mathsf{E}\Phi \stackrel{\text{def}}{\Leftrightarrow}$ there is an infinite path π from w such that $\mathcal{K}, \pi \models \Phi$.

Let $\pi = w_0, w_1, \ldots$ be an infinite path of $\mathcal{K}$. Let us define $\mathfrak{v}(\pi, \mathsf{X}^j\mathtt{x}) \stackrel{\text{def}}{=} \mathfrak{v}(w_j, \mathtt{x})$, for all terms of the form $\mathsf{X}^j\mathtt{x}$. For every n, $\pi[n, +\infty)$ is the suffix of π truncated by the n first worlds.

- $\mathcal{K}, \pi \models \mathtt{t} = \mathfrak{d} \stackrel{\text{def}}{\Leftrightarrow} \mathfrak{v}(\pi, \mathtt{t}) = \mathfrak{d}$; $\mathcal{K}, \pi \models \mathtt{t}_1 \sim \mathtt{t}_2 \stackrel{\text{def}}{\Leftrightarrow} \mathfrak{v}(\pi, \mathtt{t}_1) \sim \mathfrak{v}(\pi, \mathtt{t}_2)$ for all $\sim \in \{<, =\}$,
- $\mathcal{K}, \pi \models \Phi\mathsf{U}\Psi \stackrel{\text{def}}{\Leftrightarrow}$ there is $j \geqslant 0$ such that $\mathcal{K}, \pi[j, +\infty) \models \Psi$ and for all $j' \in [0, j-1]$, we have $\mathcal{K}, \pi[j', +\infty) \models \Phi$;
- $\mathcal{K}, \pi \models \mathsf{X}\Phi \stackrel{\text{def}}{\Leftrightarrow} \mathcal{K}, \pi[1, +\infty) \models \Phi$.

Let us define two fragments of CTL*($\mathbb{Z}$). Formulae in the logic CTL($\mathbb{Z}$) are of the form

$$\phi := \mathsf{E}\,\Theta \mid \mathsf{A}\,\Theta \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \mathsf{EX}\phi \mid \mathsf{E}\phi\mathsf{U}\phi \mid \mathsf{E}\phi\mathsf{R}\phi \mid \mathsf{AX}\phi \mid \mathsf{A}\phi\mathsf{U}\phi \mid \mathsf{A}\phi\mathsf{R}\phi,$$

where Θ is a constraint. LTL($\mathbb{Z}$) formulae are defined from path formulae for CTL*($\mathbb{Z}$) according to $\Phi := \Theta \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \mathsf{X}\Phi \mid \Phi\mathsf{U}\Phi \mid \Phi\mathsf{R}\Phi$, where Θ is a constraint. Negation occurs only in constraints since the LTL logical connectives have their dual in LTL($\mathbb{Z}$). In contrast to CTL*($\mathbb{Z}$) and CTL($\mathbb{Z}$), LTL($\mathbb{Z}$) formulae are evaluated over infinite paths of valuations $\mathfrak{v} : \mathrm{VAR} \to \mathbb{Z}$ (no branching involved).

The *satisfiability problem for* CTL*($\mathbb{Z}$), written SAT(CTL*($\mathbb{Z}$)), is defined as follows.
Input: A CTL*($\mathbb{Z}$) state formula ϕ.
Question: Is there a total Kripke structure $\mathcal{K}$ and a world w such that $\mathcal{K}, w \models \phi$?
The satisfiability problem SAT(CTL($\mathbb{Z}$)) for CTL($\mathbb{Z}$) is defined analogously; for LTL($\mathbb{Z}$), SAT(LTL($\mathbb{Z}$)) is the problem to decide whether there exists an infinite sequence of valuations $\mathfrak{v} : \mathrm{VAR} \to \mathbb{Z}$ for a given LTL($\mathbb{Z}$) formula Φ.

Decidability, and, more precisely, PSPACE-completeness of SAT(LTL($\mathbb{Z}$)) is shown in [24]. For some strict fragments of CTL*($\mathbb{Z}$), decidability is shown in [11, 12]. It is only recently in [14, 13, 15], that decidability is established for the full logic using a translation into a decidable second-order logic:

▸ **Proposition 1** ([14, 15]). SAT(CTL*($\mathbb{Z}$)) *is decidable.*

The proof in [14, 15] does not provide a complexity upper bound as the target decidable 2nd-order logic admits an automata-based decision procedure with open complexity [10, 8, 9].

Let us shortly explain why the satisfiability problem is challenging. First of all, observe that CTL*($\mathbb{Z}$) has atomic formulae in which integer values at the current and successor states are compared. This prevents us from using a simple translation from CTL*($\mathbb{Z}$) to CTL* with new propositions. Models of CTL*($\mathbb{Z}$) formulae can be viewed as an infinite network of constraints on $\mathbb{Z}$; even if a formula contains only a finite set of constants, a model may contain an infinite set of values, as it is the case for, e.g., the formula $\mathsf{EG}(\mathtt{x} < \mathsf{X}\mathtt{x})$. Hence a direct Boolean abstraction does not work. On the other hand, CTL*($\mathbb{Z}$) has no freeze quantifier and no data variable quantification, and hence no way to directly compare values at unbounded distance (but this can only be done by propagating local constraints), unlike e.g. the formalisms in [21, 63, 5, 1]. Hence, the lower bounds from [42] cannot apply either.

A problem related to satisfiability is the *model-checking problem*. Fragments of the model-checking problem involving a temporal logic similar to CTL*($\mathbb{Z}$) are investigated

in [17, 11, 12, 34] (see also [39, 20, 70, 2]). However, model-checking problems with CTL*(ℤ)-like languages are easily undecidable, see e.g. [17, Theorem 1] and [54, Theorem 4.1] (more general constraints are used in [54] but undecidability proof uses only the constraints involved herein). The difference between model-checking and satisfiability is subtle and underlines that decidability/complexity of CTL(ℤ)/CTL*(ℤ) satisfiability is not immediate.

In this paper, we prove the precise computational complexity of SAT(CTL*(ℤ)) and SAT(CTL(ℤ)). We follow the automata-based approach, that is, we translate formulae in our logics into equivalent automata – tree constraint automata for CTL(ℤ), and Rabin tree constraint automata for CTL*(ℤ) – so that we can reduce the satisfiability problem for the logics to the nonemptiness problem for the corresponding automata.

3 Tree Constraint Automata

In this section, we introduce the class of tree constraint automata that accept sets of infinite trees of the form $\mathbb{t} : [0, D-1]^* \to (\Sigma \times \mathbb{Z}^\beta)$ for some finite alphabet Σ and some $\beta \geqslant 1$. The transition relation of such automata states constraints between the β integer values at a node and the integer values at its children nodes. The acceptance condition is a Büchi condition (applied to the infinite branches of the input tree), but this can be easily extended to more general conditions (which we already consider by the end of this section). Moreover, our definition is specific to the concrete domain ℤ but it can be easily adapted to other concrete domains. Formally, a *tree constraint automaton* (TCA, for short) is a tuple $\mathbb{A} = (Q, \Sigma, D, \beta, Q_{\text{in}}, \delta, F)$, where

- Q is a finite set of locations; Σ 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),
- $Q_{\text{in}} \subseteq Q$ is the set of initial locations; $F \subseteq Q$ encodes the Büchi acceptance condition,
- δ is a *finite* subset of $Q \times \Sigma \times (\text{TreeCons}(\beta) \times Q)^D$, the transition relation. Here, TreeCons(β) denotes the constraints (Boolean combinations of atomic constraints) built over the terms $\mathtt{x}_1, \dots, \mathtt{x}_\beta, \mathtt{x}'_1, \dots, \mathtt{x}'_\beta$, where $\mathtt{x}'_i$ denotes the term $\mathsf{X}\mathtt{x}_i$. δ consists of tuples $(q, \mathtt{a}, (\Theta_0, q_0), \dots, (\Theta_{D-1}, q_{D-1}))$, where $q \in Q$ is called the source location, $q_0, \dots, q_{D-1} \in Q$, $\mathtt{a} \in \Sigma$, and $\Theta_0, \dots, \Theta_{D-1}$ are constraints.

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

- $\rho(\varepsilon) = (q_{\text{in}}, \dots)$ such that $q_{\text{in}} \in Q_{\text{in}}$;
- for every $\mathbf{n} \in [0, D-1]^*$ with $\rho(\mathbf{n}) = (q, \mathtt{a}, (\Theta_0, q_0), \dots, (\Theta_{D-1}, q_{D-1}))$, $\mathbb{t}(\mathbf{n} \cdot i) = (\mathtt{a}_i, \boldsymbol{z}_i)$, and $\rho(\mathbf{n} \cdot i)$ starts by the location q_i for all $0 \leqslant i < D$, we have $\mathbb{t}(\mathbf{n})$ of the form $(\mathtt{a}, \boldsymbol{z})$ and $\mathbb{Z} \models \Theta_i(\boldsymbol{z}, \boldsymbol{z_i})$ for all $0 \leqslant i < D$. Here, $\mathbb{Z} \models \Theta_i(\boldsymbol{z}, \boldsymbol{z_i})$ is a shortcut for $[\vec{\mathtt{x}} \leftarrow \boldsymbol{z}, \vec{\mathtt{x}'} \leftarrow \boldsymbol{z_i}] \models \Theta_i$ where $[\vec{\mathtt{x}} \leftarrow \boldsymbol{z}, \vec{\mathtt{x}'} \leftarrow \boldsymbol{z_i}]$ is a valuation $\mathfrak{v}$ on the variables $\{\mathtt{x}_j, \mathtt{x}'_j \mid j \in [1, \beta]\}$ with $\mathfrak{v}(\mathtt{x}_j) = \boldsymbol{z}(j)$ and $\mathfrak{v}(\mathtt{x}'_j) = \boldsymbol{z_i}(j)$ for all $j \in [1, \beta]$.

We show an example of a run ρ on $\mathbb{t}$ in Figure 1. Suppose ρ is a run of $\mathbb{A}$. Given a path $\pi = j_1 \cdot j_2 \cdot j_3 \dots$ in ρ starting from ε, we define $\inf(\rho, \pi)$ to be the set of locations that appear infinitely often as the source locations of the transitions in $\rho(\varepsilon)\rho(j_1)\rho(j_1 \cdot j_2)\rho(j_1 \cdot j_2 \cdot j_3) \dots$. 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 $\mathbb{t}$ that admit an accepting run.

Nonemptiness problem. As usual, the *nonemptiness problem for TCA* asks whether a TCA $\mathbb{A}$ satisfies $\mathrm{L}(\mathbb{A}) \neq \varnothing$. To define the size of $\mathbb{A}$ in a reasonably succinct encoding, we need to consider the size of constraints from TreeCons(β). Indeed, unlike (plain) Büchi tree automata [68], 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 card(δ) is a priori unbounded, even if Q and Σ are fixed. We write $\mathtt{MCS}(\mathbb{A})$ to denote the maximal size of a constraint occurring in $\mathbb{A}$ (with binary encoding of the integers). The complexity of the nonemptiness problem should take into account these parameters. Note also that our automaton model differs from the Presburger Büchi tree automata from [62, 6] for which, in the runs, arithmetical expressions are related to constraints between numbers of children labelled by different locations. Herein, the arithmetical expressions state constraints between integer values.

Next, we introduce a variant of TCA by considering the Rabin acceptance condition (as opposed to the Büchi acceptance condition). A *Rabin tree constraint automaton* (Rabin TCA, for short) is a tuple $\mathbb{A} = (Q, \Sigma, D, \beta, Q_{\mathrm{in}}, \delta, \mathcal{F})$ defined as for TCA except that $\mathcal{F}$ is a set of pairs of the form (L, U), where $L, U \subseteq Q$. All the definitions about TCA apply except that a run $\rho : [0, D-1]^* \to \delta$ is *accepting* iff for all paths π in ρ starting from ε, there is some $(L, U) \in \mathcal{F}$ such that $\inf(\rho, \pi) \cap L \neq \varnothing$ and $\inf(\rho, \pi) \cap U = \varnothing$.

Finite alphabet. The set Σ in data trees $\mathbb{t} : [0, D-1]^* \to (\Sigma \times \mathbb{Z}^\beta)$ plays no specific role herein, especially that it could be encoded with simple constraints of the form $\mathtt{x}^\star = \mathfrak{d}$, where $\mathtt{x}^\star$ is a distinguished variables. Its inclusion is more handy when the logical atomic formulae include constraints on variables *and* propositional variables, as done in [27, Section 5.2] dedicated to description logics (developments on description logics are very little in this paper, due to lack of space).

4 Complexity of the Nonemptiness Problem for TCA

This section is dedicated to prove the ExpTime-completeness of the nonemptiness problem for TCA (Theorem 11) and Rabin TCA (Theorem 13) (we make a distinction between TCA and Rabin TCA because the complexity bounds differ slightly, see Lemma 10 and Lemma 12). Before we prove the ExpTime upper bound, let us drop a few words on the lower bound. We show ExpTime-hardness of the nonemptiness problem for TCA by reduction from the acceptance problem for alternating Turing machines running in polynomial space, see e.g. [18, Corollary 3.6]. Indeed, the polynomial-space tape using a finite alphabet Σ can be encoded by a polynomial amount of variables taking values in $[1, \mathrm{card}(\Sigma)]$, details can be found in [27, Section 4.1]. ExpTime-hardness for Rabin TCA follows, as every TCA with set F of accepting locations can be encoded as a Rabin TCA with a single Rabin pair $(F, \varnothing)$.

The proof of the ExpTime upper bound is divided into two parts. In order to determine whether $\mathrm{L}(\mathbb{A})$ is nonempty for a given TCA $\mathbb{A}$, we first reduce the existence of some tree $\mathbb{t} \in \mathrm{L}(\mathbb{A})$ to the existence of some *regular symbolic tree* that is *satisfiable*, that is, it admits a concrete model (Sections 4.1 and 4.2). Second, we characterise the complexity of determining the existence of such satisfiable regular symbolic trees (Section 4.3). The result for Rabin TCA is presented in Section 4.4.

From now on, we assume a fixed TCA $\mathbb{A} = (Q, \Sigma, D, \beta, Q_{\mathrm{in}}, \delta, F)$ with the constants $\mathfrak{d}_1, \dots, \mathfrak{d}_\alpha$ occurring in $\mathbb{A}$ such that $\mathfrak{d}_1 < \cdots < \mathfrak{d}_\alpha$ (we assume there is at least one constant).

4.1 Symbolic Trees

A *type* over the variables $\mathtt{z}_1, \dots, \mathtt{z}_n$ is an expression of the form

$$(\textstyle\bigwedge_i \Theta_i^{\mathrm{CST}}) \wedge (\bigwedge_{i<j} \mathtt{z}_i \sim_{i,j} \mathtt{z}_j), \text{ where}$$

- for all $i \in [1, n]$, Θ_i^{CST} is equal to either $\mathtt{z}_i < \mathfrak{d}_1$, or $\mathtt{z}_i > \mathfrak{d}_\alpha$ or $\mathtt{z}_i = \mathfrak{d}$ for some $\mathfrak{d} \in [\mathfrak{d}_1, \mathfrak{d}_\alpha]$.

This definition goes a bit beyond the constraint language in $\mathbb{Z}$ (because of expressions of

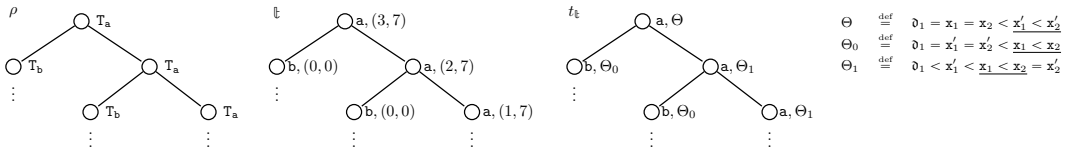


Figure 1 A tree $\mathbb{t}$ (middle), a run ρ of some TCA on $\mathbb{t}$ (left), where, $\mathtt{T_a} = (q, \mathtt{a}, (\Theta_0, q), (\Theta_1, q))$ and $\mathtt{T_b} = (q, \mathtt{b}, (\Theta_0, q), (\Theta_1, q))$, and the symbolic tree $t_{\mathbb{t}}$ (abstraction of $\mathbb{t}$) (right).

the form $\mathtt{z}_i < \mathfrak{d}_1$ and $\mathtt{z}_i > \mathfrak{d}_\alpha$), but this is harmless in the sequel. What really matters in a type is the way the variables are compared to each other and to the constants.

- $\sim_{i,j} \in \{>, =, <\}$ for all $i < j$.

Checking the satisfiability of a type can be done in polynomial-time, based on a standard cycle detection, see e.g. [17, Lemma 5.5]. The set of *satisfiable types* built over the variables $\mathtt{x}_1, \ldots, \mathtt{x}_\beta, \mathtt{x}'_1, \ldots, \mathtt{x}'_\beta$ is written $\mathrm{SatTypes}(\beta)$ (n above is equal here to 2β). Observe that $\mathrm{card}(\mathrm{SatTypes}(\beta)) \leqslant ((\mathfrak{d}_\alpha - \mathfrak{d}_1) + 3)^{2\beta} \times 3^{\beta^2}$. The *restriction* of the type Θ to some set of variables $X \subseteq \{\mathtt{x}_i, \mathtt{x}'_i \mid i \in [1, \beta]\}$ is made of all the conjuncts in which only variables in X occur. The type Θ restricted to $\{\mathtt{x}'_i \mid i \in [1, \beta]\}$ *agrees* with the type Θ' restricted to $\{\mathtt{x}_i \mid i \in [1, \beta]\}$ iff Θ and Θ' are logically equivalent modulo the renaming for which $\mathtt{x}_i$ and $\mathtt{x}'_i$ are substituted, for all $i \in [1, \beta]$. For instance, in Figure 1, Θ restricted to $\{\mathtt{x}'_1, \mathtt{x}'_2\}$ agrees with Θ_0 restricted to $\{\mathtt{x}_1, \mathtt{x}_2\}$. The main properties we use about satisfiable types are stated below.

(I) Let $\boldsymbol{z}, \boldsymbol{z}' \in \mathbb{Z}^\beta$. There is a unique $\Theta \in \mathrm{SatTypes}(\beta)$ such that $\mathbb{Z} \models \Theta(\boldsymbol{z}, \boldsymbol{z}')$.

(II) For every constraint Θ built over the variables $\mathtt{x}_1, \ldots, \mathtt{x}_\beta, \mathtt{x}'_1, \ldots, \mathtt{x}'_\beta$ and the constants $\mathfrak{d}_1, \ldots, \mathfrak{d}_\alpha$ there is a disjunction $\Theta_1 \vee \cdots \vee \Theta_\gamma$ logically equivalent to Θ and each Θ_i belongs to $\mathrm{SatTypes}(\beta)$ (empty disjunction stands for $\bot$).

(III) For all $\Theta \neq \Theta' \in \mathrm{SatTypes}(\beta)$, the constraint $\Theta \wedge \Theta'$ is not satisfiable.

The proof is by an easy verification and this justifies the term 'type' used in this context.

Abstraction with types. A *symbolic tree* t is a map $t : [0, D-1]^* \to \Sigma \times \mathrm{SatTypes}(\beta)$. Symbolic trees are intended to be abstractions of trees labelled with concrete values in $\mathbb{Z}$. Given a tree $\mathbb{t} : [0, D-1]^* \to \Sigma \times \mathbb{Z}^\beta$, its *abstraction* is the symbolic tree $t_{\mathbb{t}} : [0, D-1]^* \to \Sigma \times \mathrm{SatTypes}(\beta)$ such that for all $\mathbf{n} \cdot i \in [0, D-1]^*$ with $\mathbb{t}(\mathbf{n}) = (\mathtt{a}, \boldsymbol{z})$ and $\mathbb{t}(\mathbf{n} \cdot i) = (\mathtt{a}_i, \boldsymbol{z}_i)$, $t_{\mathbb{t}}(\mathbf{n} \cdot i) \stackrel{\text{def}}{=} (\mathtt{a}_i, \Theta_i)$ for the unique $\Theta_i \in \mathrm{SatTypes}(\beta)$ such that $\mathbb{Z} \models \Theta_i(\boldsymbol{z}, \boldsymbol{z}_i)$. Note that the primed variables in Θ_i refer to the β values at the node $\mathbf{n} \cdot i$, whereas the unprimed ones refer to the β values at the parent node $\mathbf{n}$. At the root ε with $\mathbb{t}(\varepsilon) = (\mathtt{a}, \boldsymbol{z})$, we have $t_{\mathbb{t}}(\varepsilon) \stackrel{\text{def}}{=} (\mathtt{a}, \Theta)$ for the unique $\Theta \in \mathrm{SatTypes}(\beta)$ such that $\mathbb{Z} \models \Theta(\mathbf{0}, \boldsymbol{z})$, where $\mathbf{0} \in \mathbb{Z}^\beta$ is arbitrary as there are actually no parent values at the root. A symbolic tree t is *satisfiable* $\stackrel{\text{def}}{\Leftrightarrow}$ there is $\mathbb{t} : [0, D-1]^* \to \Sigma \times \mathbb{Z}^\beta$ such that $t_{\mathbb{t}} = t$. We say that $\mathbb{t}$ *witnesses the satisfaction of* t, also written $\mathbb{t} \models t$. A symbolic tree t is *regular* if its set of subtrees is finite.

$\mathbb{A}$-consistency. In our quest to decide whether $\mathrm{L}(\mathbb{A}) \neq \varnothing$, we are interested in symbolic trees that satisfy certain properties that we subsume under the name $\mathbb{A}$-*consistent*. A symbolic tree $t : [0, D-1]^* \to \Sigma \times \mathrm{SatTypes}(\beta)$ is $\mathbb{A}$-*consistent* if the following conditions are satisfied:

- t is *locally consistent*: for every node $\mathbf{n}$, the type Θ labelling $\mathbf{n}$ restricted to $\mathtt{x}'_1, \ldots, \mathtt{x}'_\beta$ agrees with all types Θ_i labelling its children nodes $\mathbf{n} \cdot i$ restricted to $\mathtt{x}_1, \ldots, \mathtt{x}_\beta$, and
- there is an accepting run ρ of $\mathbb{A}$ (but ignoring the conditions on data values) such that for all $\mathbf{n} \in [0, D-1]^*$ with $t(\mathbf{n}) = (\mathtt{a}, \Theta)$, $t(\mathbf{n} \cdot i) = (\mathtt{a}_i, \Theta_i)$ for all $i \in [0, D-1]$, and $\rho(\mathbf{n}) = (q, \mathtt{a}, (\Theta'_0, q_0) \ldots (\Theta'_{D-1}, q_{D-1}))$, we have $\Theta_i \models \Theta'_i$ for all $i \in [0, D-1]$.

▸ **Example 2.** In Figure 1, we show a tree $\mathbb{t}$ with concrete values in $\mathbb{Z}^\beta$ for $\beta = 2$ (middle) and its abstraction $t_{\mathbb{t}}$ (right). We assume that $\mathfrak{d}_1 = 0$ is the only constant; consequently, $t_{\mathbb{t}}$ uses constraints in SatTypes(β) that are built with variables $\mathtt{x}_1, \mathtt{x}_2$, their primed variants $\mathtt{x}'_1, \mathtt{x}'_2$, and the constant $\mathfrak{d}_1$. We underline constraints to illustrate the property of local consistency. It is not hard to prove that the set of all $\mathbb{A}$-consistent symbolic trees is ω-regular, that is, it can be accepted by a classical tree automaton without constraints. In the following, we use the standard letter A to distinguish automata *without constraints* from TCA.

▸ **Lemma 3.** *There exists a Büchi tree automaton (without constraints) $A_{cons(\mathbb{A})}$ such that $\mathrm{L}(A_{cons(\mathbb{A})})$ is equal to the set of $\mathbb{A}$-consistent symbolic trees.*

The locations in $A_{\mathrm{cons}(\mathbb{A})}$ are from SatTypes(β) $\times Q$ and the transition relation for $A_{\mathrm{cons}(\mathbb{A})}$ can be decided in polynomial-time in $\mathrm{card}(\delta) + \beta + \mathrm{card}(\Sigma) + D + \mathtt{MCS}(\mathbb{A})$.

However, not every $\mathbb{A}$-consistent symbolic tree admits a concrete model. Thus the more important property is to check whether $\mathrm{L}(A_{\mathrm{cons}(\mathbb{A})})$ contains some *satisfiable* symbolic tree (and we explain how to do this in the next two subsections). The result below is a variant of many similar results relating symbolic models and concrete models in logics for concrete domains, see e.g. [23, Corollary 4.1], [38, Lemma 3.4], [16, Theorem 25], [46, Theorem 11].

▸ **Lemma 4.** $\mathrm{L}(\mathbb{A}) \neq \varnothing$ *iff there is a* satisfiable *symbolic tree in* $\mathrm{L}(A_{cons(\mathbb{A})})$.

4.2 Satisfiability for Regular Locally Consistent Symbolic Trees

Below, we focus on deciding when $\mathrm{L}(A_{\mathrm{cons}(\mathbb{A})})$ contains a *satisfiable* symbolic tree, while evaluating the complexity to check its existence. Given a locally consistent symbolic tree $t : [0, D-1]^* \to \Sigma \times \mathrm{SatTypes}(\beta)$, we introduce an infinite labelled graph that contains exactly the same types as t but expressed in a tree-like graph from which it is convenient to characterize satisfiability in terms of paths, under the premise that t is regular. Similar symbolic structures are introduced in [49, 23, 14, 46]. The graph is equal to the structure

$$G_t^{\mathrm{C}} = (V_t, \xrightarrow{=}, \xrightarrow{<}, U_{<\mathfrak{d}_1}, (U_i)_{i\in[\mathfrak{d}_1,\mathfrak{d}_\alpha]}, U_{>\mathfrak{d}_\alpha}),$$

where $V_t = [0, D-1]^* \times (\{\mathtt{x}_1, \ldots, \mathtt{x}_\beta\} \cup \{\mathfrak{d}_1, \mathfrak{d}_\alpha\})$, $\xrightarrow{=}$ and $\xrightarrow{<}$ are two binary relations over V_t, and $\{U_{<\mathfrak{d}_1}, U_{\mathfrak{d}_1}, U_{\mathfrak{d}_1+1}, \ldots, U_{\mathfrak{d}_\alpha}, U_{>\mathfrak{d}_\alpha}\}$ is a partition of V_t. Elements in $\{\mathtt{x}_1, \ldots, \mathtt{x}_\beta\} \cup \{\mathfrak{d}_1, \mathfrak{d}_\alpha\}$ are denoted by $\mathtt{xd}, \mathtt{xd}_1, \mathtt{xd}_2, \ldots$ (variables or constants). Moreover, $V_t^\beta \stackrel{\text{def}}{=} [0, D-1]^* \times \{\mathtt{x}_1, \ldots, \mathtt{x}_\beta\}$. The rationale behind the construction of G_t^{C} is to reflect the constraints between parent and children nodes as well as the constraints regarding constants, in such a way that, if $\mathbb{t}$ witnesses the satisfaction of t, then, e.g., $\mathbb{t}(\mathbf{n})(\mathtt{xd}) < \mathbb{t}(\mathbf{n}')(\mathtt{xd}')$ if $(\mathbf{n}, \mathtt{xd}) \xrightarrow{<} (\mathbf{n}', \mathtt{xd}')$, and $\mathbb{t}(\mathbf{n})(\mathtt{xd}) = \mathfrak{d}_1$ if $(\mathbf{n}, \mathtt{xd}) \in U_{\mathfrak{d}_1}$. Here are all conditions for building G_t^{C}.

(VAR) For all $(\mathbf{n}, \mathtt{x}_i), (\mathbf{n}', \mathtt{x}_{i'}) \in V_t^\beta$, for all $\sim \in \{<, =\}$, $(\mathbf{n}, \mathtt{x}_i) \xrightarrow{\sim} (\mathbf{n}', \mathtt{x}_{i'})$ iff either $\mathbf{n}' = \mathbf{n} \cdot j$ and $\mathtt{x}_i \sim \mathtt{x}'_{i'}$ in Θ with $t(\mathbf{n}') = (\mathtt{a}, \Theta)$, or $\mathbf{n} = \mathbf{n}'$ and $\mathtt{x}'_i \sim \mathtt{x}'_{i'}$ in Θ with $t(\mathbf{n}') = (\mathtt{a}, \Theta)$, or $\mathbf{n} = \mathbf{n}' \cdot j$ and $\mathtt{x}'_i \sim \mathtt{x}_{i'}$ in Θ with $t(\mathbf{n}) = (\mathtt{a}, \Theta)$.

(P1) For all $\mathfrak{d} \in [\mathfrak{d}_1, \mathfrak{d}_\alpha]$ and $(\mathbf{n}, \mathtt{x}_j) \in V_t^\beta$, $(\mathbf{n}, \mathtt{x}_j) \in U_{\mathfrak{d}}$ iff $\mathtt{x}'_j = \mathfrak{d}$ in Θ with $t(\mathbf{n}) = (\mathtt{a}, \Theta)$.

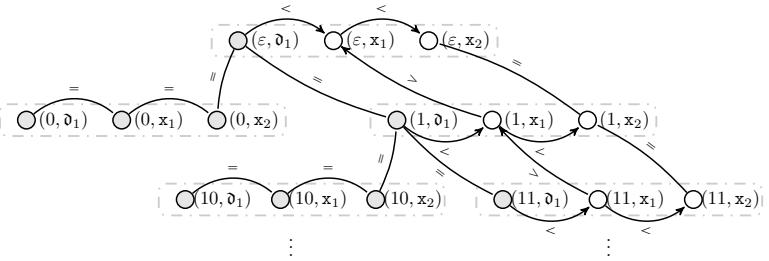
(P2) For all $(\mathbf{n}, \mathtt{x}_j) \in V_t^\beta$, $(\mathbf{n}, \mathtt{x}_j) \in U_{<\mathfrak{d}_1}$ iff $\mathtt{x}'_j < \mathfrak{d}_1$ in Θ with $t(\mathbf{n}) = (\mathtt{a}, \Theta)$.

(P3) For all $(\mathbf{n}, \mathtt{x}_j) \in V_t^\beta$, $(\mathbf{n}, \mathtt{x}_j) \in U_{>\mathfrak{d}_\alpha}$ iff $\mathtt{x}'_j > \mathfrak{d}_\alpha$ in Θ with $t(\mathbf{n}) = (\mathtt{a}, \Theta)$.

(P4) For all $\mathbf{n} \in [0, D-1]^*$, $(\mathbf{n}, \mathfrak{d}_1) \in U_{\mathfrak{d}_1}$ and $(\mathbf{n}, \mathfrak{d}_\alpha) \in U_{\mathfrak{d}_\alpha}$.

(CONS) This is about elements of V_t labelled by constants and how the edge labels reflect the relationships between the constants. Formally, for all $((\mathbf{n}, \mathtt{xd}), (\mathbf{n}', \mathtt{xd}')) \in (V_t \times V_t) \backslash (V_t^\beta \times V_t^\beta)$, for all $\mathfrak{d}^\dagger, \mathfrak{d}^{\dagger\dagger}$ in '$< \mathfrak{d}_1$', $\mathfrak{d}_1$, …, $\mathfrak{d}_\alpha$, '$> \mathfrak{d}_\alpha$' s.t. $(\mathbf{n}, \mathtt{xd}) \in U_{\mathfrak{d}^\dagger}$ and $(\mathbf{n}', \mathtt{xd}') \in U_{\mathfrak{d}^{\dagger\dagger}}$, for all $\sim \in \{<, =\}$, $(\mathbf{n}, \mathtt{xd}) \xrightarrow{\sim} (\mathbf{n}', \mathtt{xd}')$ iff either $\mathfrak{d}^\dagger, \mathfrak{d}^{\dagger\dagger} \in [\mathfrak{d}_1, \mathfrak{d}_\alpha]$ and $\mathfrak{d}^\dagger \sim \mathfrak{d}^{\dagger\dagger}$, or $\mathfrak{d}^\dagger =$ '$< \mathfrak{d}_1$', $\mathfrak{d}^{\dagger\dagger} \neq$ '$< \mathfrak{d}_1$' and $\sim$ is equal to $<$ or $\mathfrak{d}^\dagger \neq$ '$> \mathfrak{d}_\alpha$', $\mathfrak{d}^{\dagger\dagger} =$ '$> \mathfrak{d}_\alpha$' and $\sim$ is equal to $<$.

Below, we illustrate the definition of the graph $G^{\mathrm{C}}_{t_{\mathbb{t}}}$ for the symbolic tree $t_{\mathbb{t}}$ in Figure 1. The edges labelled with $=$ or $<$ reflect the constraints (we omit edges if they can be inferred from the other edges). For instance, $(1, \mathtt{x}_1) \xrightarrow{\leqslant} (\varepsilon, \mathtt{x}_1)$ corresponds to the constraint $\mathtt{x}'_1 < \mathtt{x}_1$. Grey nodes are in $U_{\mathfrak{d}_1}$, all other nodes are in $U_{>\mathfrak{d}_1}$ (no nodes in $U_{<\mathfrak{d}_1}$).



A map $p : \mathbb{N} \to V_t$ is a *path map* in $G^{\mathrm{C}}_t \overset{\text{def}}{\Leftrightarrow}$ for all $i \in \mathbb{N}$, either $p(i) \xrightarrow{=} p(i+1)$ or $p(i) \xrightarrow{\leqslant} p(i+1)$ in G^{C}_t. Similarly, $r : \mathbb{N} \to V_t$ is a *reverse path map* in $G^{\mathrm{C}}_t \overset{\text{def}}{\Leftrightarrow}$ for all $i \in \mathbb{N}$, either $r(i) \xrightarrow{=} r(i+1)$ or $r(i+1) \xrightarrow{\leqslant} r(i)$. A path map p (resp. reverse path map r) is *strict* $\overset{\text{def}}{\Leftrightarrow}$ $\{i \in \mathbb{N} \mid p(i) \xrightarrow{\leqslant} p(i+1)\}$ (resp. $\{i \in \mathbb{N} \mid r(i+1) \xrightarrow{\leqslant} r(i)\}$) is infinite. An *infinite branch* $\mathcal{B}$ is an element of $[0, D-1]^\omega$. We write $\mathcal{B}[i, j]$ with $i \leqslant j$ to denote the subsequence $\mathcal{B}(i) \cdot \mathcal{B}(i+1) \cdots \mathcal{B}(j)$. Given $(\mathbf{n}, \mathtt{xd}) \in V_t$, a path map p *from* $(\mathbf{n}, \mathtt{xd})$ *along* $\mathcal{B}$ is such that $p(0) = (\mathbf{n}, \mathtt{xd})$ and for all $i \geqslant 0$, $p(i)$ is of the form $(\mathbf{n} \cdot \mathcal{B}[0, i], \cdot)$. A reverse path map r from $(\mathbf{n}, \mathtt{xd})$ along $\mathcal{B}$ admits a similar definition. We present the condition $(\bigstar^{\mathrm{C}})$ that is the central property for characterising regular symbolic trees in $\mathrm{L}(A_{\mathrm{cons}(\mathbb{A})})$ that are satisfiable, following the remarkable result established in [46, Lemma 22] that non-satisfiability of a symbolic tree can be witnessed along a *single branch*.

$(\bigstar^{\mathrm{C}})$ There are *no* elements $(\mathbf{n}, \mathtt{xd})$, $(\mathbf{n}, \mathtt{xd}')$ in G^{C}_t (same node $\mathbf{n}$ from $[0, D-1]^*$) and no infinite branch $\mathcal{B}$ such that

1. there exists a path map p from $(\mathbf{n}, \mathtt{xd})$ along $\mathcal{B}$,
2. there exists a reverse path map r from $(\mathbf{n}, \mathtt{xd}')$ along $\mathcal{B}$,
3. p or r is strict, and
4. for all $i \in \mathbb{N}$, $p(i) \xrightarrow{\leqslant} r(i)$.

The following proposition states a key property: non-satisfaction of a regular locally consistent symbolic tree can be witnessed along a *single* branch by violation of $(\bigstar^{\mathrm{C}})$.

▸ **Proposition 5.** *For every regular locally consistent symbolic tree t, G^C_t satisfies $(\bigstar^{C})$ iff t is satisfiable.*

A proof can be found in [27, Section 7].

▸ **Example 6.** Assume that every node along the rightmost branch in the symbolic tree $t_{\mathbb{t}}$ in Figure 1 is labelled with $(\mathtt{a}, \Theta_1)$. Then $t_{\mathbb{t}}$ is not satisfiable: in order to satisfy Θ_1's conjunct $\mathtt{x}'_1 < \mathtt{x}_1$, the value of $\mathtt{x}_1$ must inevitably become finally smaller than $\mathfrak{d}_1$, violating the conjunct $\mathfrak{d}_1 < \mathtt{x}_1$. Consequently, the rightmost branch of $G^{\mathrm{C}}_{t_{\mathbb{t}}}$ presented above does not satisfy $(\bigstar^{\mathrm{C}})$: there exists a path map p from $(\varepsilon, \mathfrak{d}_1)$ along 1^ω, there exists a strict reverse path map r from $(\varepsilon, \mathtt{x}_1)$ along 1^ω, and for all $i \in \mathbb{N}$ we have $p(i) \xrightarrow{\leqslant} r(i)$.

New constant nodes. Proposition 5 above is a variant of [46, Lemma 22]. Before going any further, let us in short explain the improvement of our developments compared to what is done in [46, 45]. The *framified constraint graphs* defined in [46, Definition 14] correspond to the above defined graph G^{C}_t without $[0, D-1]^* \times \{\mathfrak{d}_1, \mathfrak{d}_\alpha\}$ and corresponding edges. However, Example 6 illustrates the importance of taking into account these elements when deciding satisfiability (without $\mathfrak{d}_1$, the graph would satisfy $(\bigstar^{\mathrm{C}})$). Actually, Example 6 invalidates $(\bigstar)$ as used in [46, 45] because the constants are missing to apply properly [15]. The problematic part in [46, 45] is due to the proof of [45, Lemma 5.18] whose main argument takes advantage

of [15] but without the elements related to constant values (see also [24, Lemma 8]). With Proposition 5, we also propose a proof to characterise satisfiability of symbolic trees that is independent of [15]. Note also that the condition ($\bigstar$) in [46, Section 3.3] generalises the condition $C_{\mathbb{Z}}$ from [23, Section 6] (see also the condition $\mathcal{C}$ in [24, Definition 2] and a similar condition in [32, Section 2]). A condition similar to ($\bigstar$) is also introduced recently in [7, Lemma 18] to decide a realizability problem based on LTL$(\mathbb{Z}, <, =)$.

We recall that there are *nonregular* locally consistent symbolic trees t such that G_t^C satisfies ($\bigstar^C$) (see e.g. [23, 46]) but t is not satisfiable; indeed, satisfiability of symbolic trees is not an ω-regular property. The next result states that ($\bigstar^C$) is ω-regular; hence, satisfiability of symbolic trees can be overapproximated advantageously.

▸ **Lemma 7.** *There is a Rabin tree automaton $A_{\bigstar^C}$ such that* $\mathrm{L}(A_{\bigstar^C}) = \{t \mid G_t^C \text{ satisfies } (\bigstar^C)\}$, *the number of Rabin pairs is bounded above by* $8(\beta+2)^2+3$, *the number of locations is exponential in* β, *the transition relation can be decided in polynomial-time in*

$$\max(\lceil log(|\mathfrak{d}_1|)\rceil, \lceil log(|\mathfrak{d}_\alpha|)\rceil) + \beta + card(\Sigma) + D.$$

Proof sketch. The proof of Lemma 7 is structured as follows (see [27, Section 4.3]). (1) We construct a Büchi word automaton A_B accepting the complement of ($\bigstar^C$) for $D = 1$. (2) A_B is nondeterministic, but we can determinize it and get a deterministic Rabin word automaton $A_{B\to R}$ such that $\mathrm{L}(A_B) = \mathrm{L}(A_{B\to R})$ (using the determinisation construction from [60, Theorem 1.1]). (3) By an easy construction, we obtain a deterministic Street word automaton A_S accepting the complement of $\mathrm{L}(A_{B\to R})$; it accepts words that satisfy ($\bigstar^C$) for $D = 1$. (4) By [60, Lemma 1.2], we construct a deterministic Rabin word automaton A_R s.t. $\mathrm{L}(A_S) = \mathrm{L}(A_R)$. (5) Finally, we construct a Rabin tree automaton $A_{\bigstar^C}$, the intuitive idea is to "let run the automaton A_R" along every branch of a run of $A_{\bigstar^C}$, doable thanks to the determinism of A_R. Since ($\bigstar^C$) states a property on every branch, we are done.

Differences with [46]. Lemma 7 is similar to [46, Proposition 26] but there is an essential difference: the number of Rabin pairs in Lemma 7 is not a constant but a value depending on β, an outcome of our investigations. It is important to know the number of Rabin pairs in $A_{\bigstar^C}$ for our complexity analysis as checking nonemptiness of Rabin tree automata is *exponential* in the number of Rabin pairs [30, Theorem 4.1]. Our proof of Lemma 7 also proposes a slight novelty compared to the construction in [46]: we design $A_{\bigstar^C}$ without firstly constructing *a tree automaton for the complement language* (as done in [46]) and then using results from [55] (elimination of alternation in tree automata). Our new approach shall be rewarding: not only we can better understand how to express the condition ($\bigstar^C$), but also we control the size parameters of $A_{\bigstar^C}$ involved in our forthcoming complexity analysis. Furthermore, it may be useful to implement the decision procedure for solving the satisfiability problem for CTL$(\mathbb{Z})$ (resp. for CTL*$(\mathbb{Z})$). Note also that the above analysis about the number of Rabin pairs is independent from the question discussed above about having the elements in $[0, D-1]^* \times \{\mathfrak{d}_1, \mathfrak{d}_\alpha\}$ within G_t^C.

Summarizing the developments so far, we can conclude this subsection as follows:

▸ **Lemma 8.** $\mathrm{L}(\mathbb{A}) \neq \varnothing$ *iff* $\mathrm{L}(A_{cons(\mathbb{A})}) \cap \mathrm{L}(A_{\bigstar^C}) \neq \varnothing$.

For its proof, by way of example, if $\mathrm{L}(A_{\mathrm{cons}(\mathbb{A})}) \cap \mathrm{L}(A_{\bigstar^C})$ is non-empty, then as $\mathrm{L}(A_{\mathrm{cons}(\mathbb{A})}) \cap \mathrm{L}(A_{\bigstar^C})$ is regular, it contains a regular $\mathbb{A}$-consistent symbolic tree t (see e.g. [58] and [64, Section 6.3] for the existence of regular trees) and by Proposition 5, t is satisfiable. By Lemma 4, we get $\mathrm{L}(\mathbb{A}) \neq \varnothing$. For the other direction, we use Lemma 3 as well as the property that for every satisfiable symbolic tree t, G_t^C satisfies the condition ($\bigstar^C$).

4.3 ExpTime Upper Bound for TCAs

Lemma 8 justifies why deciding the nonemptiness of $\mathrm{L}(A_{\mathrm{cons}(\mathbb{A})}) \cap \mathrm{L}(A_{\bigstar C})$ is crucial. In the proof of Lemma 9 below (see [27, Section 4.4]), we propose a construction for the intersection of Rabin tree automata that only performs an exponential blow-up for the number of locations, which is fine for our purposes.

▸ **Lemma 9.** *There is a Rabin tree automaton A such that $\mathrm{L}(A) = \mathrm{L}(A_{cons(\mathbb{A})}) \cap \mathrm{L}(A_{\bigstar C})$ and the number of Rabin pairs is polynomial in β, the number of locations is in $\mathcal{O}(card(\mathrm{SatTypes}(\beta)) \times card(Q) \times 2^{P(\beta)})$ for some polynomial $P(\cdot)$ and the transition relation can be decided in polynomial-time in $card(\delta) + \beta + card(\Sigma) + D + \mathtt{MCS}(\mathbb{A})$.*

Nonemptiness of Rabin tree automata is polynomial in the cardinality of the transition relation and exponential in the number of Rabin pairs, see e.g. [30, Theorem 4.1]. More precisely, it is in time $(m \times n)^{\mathcal{O}(n)}$, where m is the number of locations and n is the number of Rabin pairs, see the statement [30, Theorem 4.1]. However, this is not exactly what we need herein, as the complexity expression above concerns *binary* trees, and it assumes that the transition relation δ can be decided in constant time. If, as in our case, $D \geqslant 1$ and deciding whether a tuple belongs to δ requires γ time units, checking nonemptiness is actually in time $(\mathrm{card}(\delta) \times \gamma \times n)^{\mathcal{O}(n)}$ (by scrutiny of the proof of [30, Theorem 4.1], page 144). Here, γ may depend on parameters related to $\mathbb{A}$ and in Lemma 10 below, γ takes the value $\mathrm{card}(\delta) + \beta + \mathrm{card}(\Sigma) + D + \mathtt{MCS}(\mathbb{A})$ (by Lemma 9). Hence the following result:

▸ **Lemma 10.** *The nonemptiness problem for TCA can be solved in time in $\mathcal{O}(R_1\big(card(Q) \times card(\delta) \times \mathtt{MCS}(\mathbb{A}) \times card(\Sigma) \times R_2(\beta)\big)^{R_2(\beta) \times R_3(D)})$ for some polynomials R_1, R_2 and R_3.*

Assuming that the size of the TCA $\mathbb{A} = (Q, \Sigma, D, \beta, Q_{\mathrm{in}}, \delta, F)$, written $\mathtt{size}(\mathbb{A})$, is polynomial in $\mathrm{card}(Q) + \mathrm{card}(\delta) + D + \beta + \mathtt{MCS}(\mathbb{A})$ (which makes sense for a reasonably succinct encoding), from the computation of the bound in Lemma 10, the nonemptiness of $\mathrm{L}(\mathbb{A})$ can be checked in time $\mathcal{O}(R(\mathtt{size}(\mathbb{A}))^{R'(\beta+D)})$ for some polynomials R and R'. The ExpTime upper bound of the nonemptiness problem for TCA is now a consequence of the above complexity expression.

▸ **Theorem 11.** *Nonemptiness problem for tree constraint automata is ExpTime-complete.*

4.4 Rabin Tree Constraint Automata

We can prove the ExpTime upper bound of the nonemptiness problem for Rabin TCA (Theorem 13) and follow the same lines of arguments as for TCA. Given a Rabin TCA $\mathbb{A} = (Q, \Sigma, D, \beta, Q_{\mathrm{in}}, \delta, \mathcal{F})$, we define a Rabin tree automaton $A'_{\mathrm{cons}(\mathbb{A})}$ such that $\mathrm{L}(\mathbb{A}) \neq \varnothing$ iff there is $t \in \mathrm{L}(A'_{\mathrm{cons}(\mathbb{A})})$ that is satisfiable (cf. Lemma 4 for TCA). Moreover, we take advantage of $A_{\bigstar C}$ so that $\mathrm{L}(\mathbb{A}) \neq \varnothing$ iff $\mathrm{L}(A'_{\mathrm{cons}(\mathbb{A})}) \cap \mathrm{L}(A_{\bigstar C})$ is non-empty (cf. Lemma 8). It remains to determine the cost for testing nonemptiness of $\mathrm{L}(A'_{\mathrm{cons}(\mathbb{A})}) \cap \mathrm{L}(A_{\bigstar C})$. Here is the counterpart of Lemma 9 (same kind of arguments).

▸ **Lemma 12.** *There is a Rabin tree automaton A s.t. $\mathrm{L}(A) = \mathrm{L}(A'_{cons(\mathbb{A})}) \cap \mathrm{L}(A_{\bigstar C})$, the number of Rabin pairs is polynomial in $\beta + card(\mathcal{F})$, the number of locations is in $\mathcal{O}(card(\mathrm{SatTypes}(\beta)) \times card(Q) \times 2^{P(\beta + card(\mathcal{F}))})$ for some polynomial $P(\cdot)$, and the transition relation can be decided in polynomial-time in $card(\delta) + \beta + card(\Sigma) + D + \mathtt{MCS}(\mathbb{A})$.*

As for Lemma 10, we conclude that the nonemptiness problem for Rabin TCA can be solved in time $\mathcal{O}(R_1\big(\mathrm{card}(Q) \times \mathrm{card}(\delta) \times \mathtt{MCS}(\mathbb{A}) \times \mathrm{card}(\Sigma) \times R_2(\beta + \mathrm{card}(\mathcal{F}))\big)^{R_2(\beta + \mathrm{card}(\mathcal{F})) \times R_3(D)})$ for polynomials R_1, R_2 and R_3. The nonemptiness problem for Rabin TCA is also in ExpTime.

▸ **Theorem 13.** *The nonemptiness problem for Rabin TCA is* ExpTime*-complete.*

This result is mainly useful to characterize the complexity of SAT(CTL*(ℤ)) in Section 6.

5 Tree Constraint Automata for CTL(ℤ)

Below, we harvest the first results from what is achieved in the previous section: SAT(CTL(ℤ)) is in ExpTime. So, enriching the CTL models with numerical values interpreted in ℤ does not cause a complexity blow-up. We follow the automata-based approach and (after proving a refined version of the tree model property for CTL(ℤ)) the key step is to translate CTL(ℤ) formulae into equivalent TCA. Theorem 14 below is one of our main results.

▸ **Theorem 14.** *The satisfiability problem for* CTL(ℤ) *is* ExpTime*-complete.*

Sketch. ExpTime-hardness is inherited from CTL. For ExpTime-easiness, let ϕ be a CTL(ℤ) formula. A first step is to preprocess the formula into a formula in *simple form* (see definition in Section 2.2). Then, we can construct from a formula ϕ in simple form a TCA $\mathbb{A}_\phi$ s.t. ϕ is satisfiable iff $\mathrm{L}(\mathbb{A}_\phi) \neq \varnothing$ and $\mathbb{A}_\phi$ satisfies the following properties.

- The degree D and the number of variables β are bounded by $\mathtt{size}(\phi)$.
- The number of locations is bounded by $(D \times 2^{\mathtt{size}(\phi)}) \times (\mathtt{size}(\phi) + 1)$.
- The number of transitions is in $\mathcal{O}(2^{P(\mathtt{size}(\phi))})$ for some polynomial $P(\cdot)$.
- The finite alphabet Σ in $\mathbb{A}_\phi$ is unary; $\mathtt{MCS}(\mathbb{A}_\phi)$ is quadratic in $\mathtt{size}(\phi)$.

By Lemma 10, the nonemptiness problem for TCA can be solved in time

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

Since the transition relations of the automata $A_{\mathrm{cons}(\mathbb{A})}$ and $A_{\bigstar c}$ can be built in polynomial-time, we get that nonemptiness of $\mathrm{L}(\mathbb{A}_\phi)$ can be solved in exponential-time. ◂

Let ℕ be the concrete domain $(\mathbb{N}, <, =, (=_\mathfrak{d})_{\mathfrak{d}\in\mathbb{N}})$ for which we can also show that nonemptiness of TCA with constraints interpreted on ℕ has the same complexity as for TCA with constraints interpreted on ℤ. Let CTL(ℕ) be the variant of CTL(ℤ) with constraints interpreted on ℕ. As a corollary, SAT(CTL(ℕ)) is ExpTime-complete. With the concrete domain $(\mathbb{Q}, <, =, (=_\mathfrak{d})_{\mathfrak{d}\in\mathbb{Q}})$, all the trees in $\mathrm{L}(A_{\mathrm{cons}(\mathbb{A})})$ are satisfiable (no need to intersect $A_{\mathrm{cons}(\mathbb{A})}$ with a hypothetical $A_{\bigstar c}$, see e.g. [49, 4, 23, 38]), and therefore SAT(CTL(ℚ)) is in ExpTime too. TCA can be also used to show that the concept satisfiability w.r.t. general TBoxes for the description logic $\mathcal{ALCF}^{\mathcal{P}}(\mathbb{Z}_c)$ is in ExpTime [46, 45], see more details in [27, Section 5.2].

6 Complexity of the Satisfiability Problem for the Logic CTL*(ℤ)

We show that SAT(CTL*(ℤ)) can be solved in 2ExpTime. We follow the automata-based approach for CTL*, see e.g. [31, 30], but adapted to Rabin TCA. The main challenge here is to carefully check that essential steps for CTL* can be lifted to CTL*(ℤ), but also that computationally we are in a position to provide an optimal complexity upper bound.

Let us explain in short all steps necessary to obtain the result. We start by establishing a special form for CTL*(ℤ) formulae from which Rabin TCA will be defined, following ideas from [31] for CTL*. A CTL*(ℤ) state formula ϕ is in *special form* if it has the form below

$$\mathsf{E}\ (\mathtt{x} = 0)\ \wedge\ \big(\textstyle\bigwedge_{i\in[1,D-1]} \mathsf{AGE}\ \Phi_i\big)\ \wedge\ \big(\textstyle\bigwedge_{j\in[1,D']} \mathsf{A}\ \Phi'_j\big),$$

where the Φ_i's and the Φ'_j's are LTL(ℤ) formulae in simple form (see Section 2), for some $D \geqslant 1$, $D' \geqslant 0$. We can restrict ourselves to CTL*(ℤ) state formulae in special form (see the proof of [27, Proposition 6]).

▸ **Proposition 15.** *For every* CTL*$(\mathbb{Z})$ *formula* ϕ*, one can construct in polynomial time in the size of* ϕ *a* CTL*$(\mathbb{Z})$ *formula* ϕ' *in special form s.t.* ϕ *is satisfiable iff* ϕ' *is satisfiable.*

So ϕ' is also of polynomial size in the size of ϕ. Let us state a tree model property of special formulae, with a strict discipline on the witness paths. Proposition 16 below is a counterpart of [31, Theorem 3.2] but for CTL*$(\mathbb{Z})$ instead of CTL*, see also the variant [38, Lemma 3.3].

▸ **Proposition 16.** *Let* ϕ *be a* CTL*$(\mathbb{Z})$ *formula in special form built over* $\mathtt{x}_1, \dots, \mathtt{x}_\beta$*.* ϕ *is satisfiable iff there is a tree* $\mathbb{t} : [0, D-1] \to \mathbb{Z}^\beta$ *such that* $\mathbb{t}, \varepsilon \models \phi$ *and for each* $i \in [1, D-1]$*,* $\mathbb{t}$ *satisfies* $\mathsf{AGE}\ \Phi_i$ *via* i*, that is, if* $\mathbb{t}, \mathbf{n} \models \mathsf{E}\ \Phi_i$*, then* Φ_i *is satisfied on the path* $\mathbf{n} \cdot i \cdot 0^\omega$*.*

Proposition 16 justifies our restriction to infinite trees and to TCA in the rest of this section. Proposition 15 allows us to restrict our attention to constructing automata for formulae of (only) the form $\mathsf{AGE}\ \Phi$ and $\mathsf{A}\ \Phi$, where Φ is a simple formula in LTL$(\mathbb{Z})$. The first step is to translate simple formulae in LTL$(\mathbb{Z})$ into equivalent *word* constraint automata (TCA with degree $D = 1$). Adapting the standard automata-based approach for LTL [69], we can show the following proposition (see the proof of [27, Proposition 8]).

▸ **Proposition 17.** *Let* Φ *be an* LTL$(\mathbb{Z})$ *formula in simple form. There is a constraint word automaton* $\mathbb{A}_\Phi$ *such that* $\{\mathfrak{w} : \mathbb{N} \to \mathbb{Z}^\beta \mid \mathfrak{w} \models \Phi\} = \mathrm{L}(\mathbb{A}_\Phi)$*, and the following conditions hold.*
(I) *The number of locations in* $\mathbb{A}_\Phi$ *is bounded by* $\mathtt{size}(\Phi) \times 2^{2 \times \mathtt{size}(\Phi)}$*.*
(II) *The cardinality of* δ *in* $\mathbb{A}_\Phi$ *is in* $\mathcal{O}(2^{P(\mathtt{size}(\Phi))})$ *for some polynomial* $P(\cdot)$*.*
(III) *The maximal size of a constraint in* $\mathbb{A}_\Phi$ *is quadratic in* $\mathtt{size}(\Phi)$*.*

We can now construct, for every $i \in [0, D-1]$, a TCA $\mathbb{A}_i$ such that $\mathrm{L}(\mathbb{A}_i) = \{\mathbb{t} : [0, D-1]^* \to \mathbb{Z}^\beta \mid \mathbb{t} \models \mathsf{AGE}\ \Phi_i$ and $\mathbb{t}$ satisfies $\mathsf{AGE}\ \Phi_i$ via $i\}$. The idea is to construct $\mathbb{A}_i$ so that it starts off the word constraint automaton $\mathbb{A}_{\Phi_i}$ at each node $\mathbf{n}$ of the tree and runs it down the designated path $\mathbf{n} \cdot i \cdot 0^\omega$ to check whether Φ_i actually holds along this path. This can be easily done for $\mathsf{AGE}\ \Phi_i$; however, for formulas of the form $\mathsf{A}\ \Phi'_j$, for this construction to be correct, the underlying constraint word automaton $\mathbb{A}'_j$ must be *deterministic*, that is, for all locations s, letters $\mathtt{a}$ and pairs of valuations $(\boldsymbol{z}, \boldsymbol{z}') \in \mathbb{Z}^{2\beta}$, there exists in $\mathbb{A}'_j$ *at most* a single transition $(s, \mathtt{a}, \Theta, s')$ such that $\mathbb{Z} \models \Theta(\boldsymbol{z}, \boldsymbol{z}')$. A well-known construction to transform nondeterministic Büchi automata to equivalent deterministic Rabin automata is due to Safra [60, Theorem 1.1]. An important step towards the optimal complexity for CTL*$(\mathbb{Z})$ is to show that it is possible to lift this construction to word constraint automata, which is a result of its own interest. A special attention is given to the cardinality of the transition relation and to the size of the constraints in transitions, as these two parameters are, a priori, unbounded in constraint automata but essential to perform a forthcoming complexity analysis.

▸ **Theorem 18.** *Let* $\mathbb{A} = (Q, \Sigma, \beta, Q_{in}, \delta, F)$ *be a Büchi word constraint automaton involving the constants* $\mathfrak{d}_1, \dots, \mathfrak{d}_\alpha$*. There is a deterministic Rabin word constraint automaton* $\mathbb{A}' = (Q', \Sigma, \beta, Q'_{in}, \delta', \mathcal{F})$ *such that* $\mathrm{L}(\mathbb{A}) = \mathrm{L}(\mathbb{A}')$ *verifying the following quantitative properties.*
(I) *card*(Q') *is exponential in card*(Q) *and the number of Rabin pairs in* $\mathbb{A}'$ *is bounded by* $2 \cdot$ *card*(Q) *(same bounds as in [60, Theorem 1.1]).*
(II) *The constraints in the transitions are from* SatTypes(β)*, are of size cubic in* $\beta + \max(\lceil log(|\mathfrak{d}_1|)\rceil, \lceil log(|\mathfrak{d}_\alpha|)\rceil)$ *and card*$(\delta') \leqslant$ *card*$(Q')^2 \times$ *card*$(\Sigma) \times ((\mathfrak{d}_\alpha - \mathfrak{d}_1) + 3)^{2\beta} \times 3^{\beta^2}$*.*

This and Proposition 17 lead us to the result below on LTL$(\mathbb{Z})$ formulae in simple form.

▸ **Corollary 19.** *Let* Φ *be an* LTL$(\mathbb{Z})$ *formula in simple form built over the variables* $\mathtt{x}_1, \dots, \mathtt{x}_\beta$ *and the constants* $\mathfrak{d}_1, \dots, \mathfrak{d}_\alpha$*. There exists a deterministic Rabin word constraint automaton* $\mathbb{A}_\Phi$ *such that* $\{\mathfrak{w} : \mathbb{N} \to \mathbb{Z}^\beta \mid \mathfrak{w} \models \Phi\} = \mathrm{L}(\mathbb{A}_\Phi)$*, and the following conditions hold.*

(I) *The number of locations in* $\mathbb{A}_\Phi$ *is bounded by* $2^{2^{P^\dagger(\mathtt{size}(\Phi))}}$ *for some polynomial* $P^\dagger(\cdot)$.
(II) *The number of Rabin pairs is bounded by* $2 \times \mathtt{size}(\Phi) \times 2^{2\times\mathtt{size}(\Phi)}$.
(III) *The cardinality of* δ *in* $\mathbb{A}_\Phi$ *is bounded by card*$(\mathrm{SatTypes}(\beta)) \times 2^{2^{P^\dagger(\mathtt{size}(\Phi))+1}}$.
(IV) $\mathtt{MCS}(\mathbb{A}_\Phi)$ *is cubic in* $\beta + \max(\lceil log(|\mathfrak{d}_1|)\rceil, \lceil log(|\mathfrak{d}_\alpha|)\rceil)$, *i.e. polynomial in* $\mathtt{size}(\Phi)$.

This enables us to use the idea illustrated above for formulas of the form $\mathsf{AGE}\ \Phi_i$ also for formulas of the form $\mathsf{A}\ \Phi'_j$, and define Rabin TCA $\mathbb{A}'_j$ such that $\mathrm{L}(\mathbb{A}'_j) = \{\mathbb{t} : [0, D-1]^* \to \mathbb{Z}^\beta \mid \mathbb{t} \text{ satisfies } \mathsf{A}\ \Phi'_j\}$. We are now ready to perform the final step towards the main result of this section. Let us recapitulate what we have so far.

- One can define a TCA $\mathbb{A}_0$ with two locations such that $\mathrm{L}(\mathbb{A}_0)$ is the set of trees $\mathbb{t} : [0, D-1]^* \to \mathbb{Z}^\beta$ such that $\mathbb{t}(\varepsilon)(\mathtt{x}_1) = 0$, to handle $\mathsf{E}(\mathtt{x}_1 = 0)$ in formulae in special form.
- For all $1 \leqslant i < D$, there are (Büchi) TCA $\mathbb{A}_i$ such that $\mathrm{L}(\mathbb{A}_i)$ is the set of trees $\mathbb{t} : [0, D-1]^* \to \mathbb{Z}^\beta$ such that $\mathbb{t}, \varepsilon \models \mathsf{AGE}\ \Phi_i$ and $\mathbb{t}$ satisfies $\mathsf{AGE}\ \Phi_i$ via i. Recall that TCA can be seen as Rabin TCA with a single Rabin pair.
- For all $1 \leqslant j \leqslant D'$, there are Rabin TCA $\mathbb{A}'_j$ such that $\mathrm{L}(\mathbb{A}'_j)$ is the set of trees $\mathbb{t}$ such that $\mathbb{t}$ satisfies $\mathsf{A}\ \Phi_j$, with an exponential number of Rabin pairs in $\mathtt{size}(\Phi)$.

To define a Rabin TCA $\mathbb{A}$ such that $\mathrm{L}(\mathbb{A}) = \mathrm{L}(\mathbb{A}_0) \bigcap_{i\in[1,D-1]} \mathrm{L}(\mathbb{A}_i) \bigcap_{j\in[1,D']} \mathrm{L}(\mathbb{A}'_j)$, and then use the complexity bounds previously established, we need the result below (see the full proof in [27, Section 6.5]).

▸ **Lemma 20.** *Let* $(\mathbb{A}_k)_{1\leqslant k\leqslant n}$ *be a family of Rabin TCA such that* $\mathbb{A}_k = (Q_k, \Sigma, D, \beta, Q_{k,in}, \delta_k, \mathcal{F}_k)$, *card*$(\mathcal{F}_k) = N_k$ *and* $N = \prod\limits_k N_k$. *There is a Rabin TCA* $\mathbb{A}$ *such that* $\mathrm{L}(\mathbb{A}) = \bigcap_k \mathrm{L}(\mathbb{A}_k)$ *and*

- *the number of Rabin pairs is equal to* N; $\mathtt{MCS}(\mathbb{A}) \leqslant n + \mathtt{MCS}(\mathbb{A}_1) + \cdots + \mathtt{MCS}(\mathbb{A}_n)$,
- *the number of locations (resp. transitions) is less than* $\big(\prod\limits_k card(Q_k)\big)(2n)^N$ *(resp.* $\prod\limits_k card(\delta_k)$*).*

Putting all results together, the nonemptiness of $\mathrm{L}(\mathbb{A})$ can be checked in double-exponential time in $\mathtt{size}(\phi)$, leading to Theorem 21 below, which is the main result of the paper. It answers open questions from [11, 15, 16, 46].

▸ **Theorem 21.** $\mathrm{SAT}(\mathrm{CTL}^*(\mathbb{Z}))$ *is 2ExpTime-complete.*

2ExpTime-hardness is from $\mathrm{SAT}(\mathrm{CTL}^*)$ [66, Theorem 5.2]. As a corollary, $\mathrm{SAT}(\mathrm{CTL}^*(\mathbb{N}))$ is also 2ExpTime-complete. Furthermore, assuming that $<_{\mathrm{pre}}$ is the prefix relation on $\{0,1\}^*$, we can use the reduction from [22, Section 4.2] to conclude $\mathrm{SAT}(\mathrm{CTL}^*(\{0,1\}^*, <_{\mathrm{pre}}))$ is 2ExpTime-complete too. Furthermore, as observed earlier, when the concrete domain is $(\mathbb{Q}, <, =, (=_\mathfrak{d})_{\mathfrak{d}\in\mathbb{Q}})$, all the trees in $\mathrm{L}(A_{\mathrm{cons}(\mathbb{A})})$ are satisfiable, and therefore $\mathrm{SAT}(\mathrm{CTL}^*(\mathbb{Q}))$ is also in 2ExpTime, which is already known from [38, Theorem 4.3].

7 Concluding Remarks

We developed an automata-based approach to solve $\mathrm{SAT}(\mathrm{CTL}(\mathbb{Z}))$ and $\mathrm{SAT}(\mathrm{CTL}^*(\mathbb{Z}))$, by introducing tree constraint automata that accept infinite data trees with data domain $\mathbb{Z}$. The nonemptiness problem for tree constraint automata with Büchi acceptance conditions (resp. with Rabin pairs) is ExpTime-complete, see Theorem 11 (resp. Theorem 13). The difficult part consists in proving the ExpTime-easiness for which we show how to substantially adapt the material in [45, Section 5.2] that guided us to design the correctness proof of $(\bigstar^{\mathrm{C}})$. The work [46] was indeed a great inspiration but we adjusted a few statements from there (see also [27]). We recall that $(\bigstar)$ in [46] is not fully correct (see Section 4.2) as we need to add constants (leading to the variant condition $(\bigstar^{\mathrm{C}})$). Moreover, our construction of the automaton in Lemma 7 does depend on the number of variables unlike [46, Proposition 26].

This is crucial for complexity, as it is related to the number of Rabin pairs. We also use [30] more precisely than [46, p.621] as we handle non-binary trees. In short, we introduced TCA for which we characterise complexity of the non-emptiness problem (providing a few improvements to [46]). We left aside the question of the expressiveness of TCA, which is interesting but out of the scope of this paper.

This lead us to show that SAT(CTL($\mathbb{Z}$)) is ExpTime-complete (Theorem 14), and SAT(CTL*($\mathbb{Z}$)) is 2ExpTime-complete (Theorem 21). The only decidability proof for SAT(CTL*($\mathbb{Z}$)) done so far, see [15, Theorem 32], is by reduction to a decidable second-order logic. Our complexity characterisation for SAT(CTL*($\mathbb{Z}$)) provides an answer to several open problems related to CTL*($\mathbb{Z}$) fragments, see e.g. [11, 38, 15, 16, 46]. We believe that our results on TCA can help to establish complexity results for other logics (see also Section 6 about a domain for strings and [33, Section 4] to handle more concrete domains).

References

1. S. Abriola, D. Figueira, and S. Figueira. Logics of repeating values on data trees and branching counter systems. In *FoSSaCS'17*, volume 10203 of *LNCS*, pages 196–212, 2017.
2. E.G. Amparore, S. Donatelli, and F. Gallà. A CTL* model checker for Petri nets. In *Petri Nets'20*, volume 12152 of *LNCS*, pages 403–413. Springer, 2020.
3. F. Baader and J. Rydval. Using model theory to find decidable and tractable description logics with concrete domains. *JAR*, 66(3):357–407, 2022.
4. Ph. Balbiani and J.F. Condotta. Computational complexity of propositional linear temporal logics based on qualitative spatial or temporal reasoning. In *FroCoS'02*, volume 2309 of *LNAI*, pages 162–173. Springer, 2002.
5. K. Bartek and M. Lelyk. Modal mu-calculus with atoms. In *CSL'17*, pages 30:1–30:21. Leibniz-Zentrum für Informatik, LIPICS, 2017.
6. B. Bednarczyk and O. Fiuk. Presburger Büchi tree automata with applications to logics with expressive counting. In *WoLLIC'22*, volume 13468 of *LNCS*, pages 295–308. Springer, 2022.
7. A. Bhaskar and M. Praveen. Realizability problem for constraint LTL. In *TIME'22*, volume 247 of *LIPIcs*, pages 8:1–8:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.
8. M. Bojańczyk. A bounding quantifier. In *CSL'04*, volume 3210 of *LNCS*, pages 41–55. Springer, 2004.
9. M. Bojańczyk and Th. Colcombet. Bounds in ω-Regularity. In *LiCS'06*, pages 285–296. IEEE Computer Society, 2006.
10. M. Bojańczyk and S. Toruńczyk. Weak MSO+U over infinite trees. In *STACS'12*, LIPIcs, pages 648–660, 2012.
11. L. Bozzelli and R. Gascon. Branching-time temporal logic extended with Presburger constraints. In *LPAR'06*, volume 4246 of *LNCS*, pages 197–211. Springer, 2006.
12. L. Bozzelli and S. Pinchinat. Verification of gap-order constraint abstractions of counter systems. *TCS*, 523:1–36, 2014.
13. C. Carapelle. *On the satisfiability of temporal logics with concrete domains*. PhD thesis, Leipzig University, 2015.
14. C. Carapelle, A. Kartzow, and M. Lohrey. Satisfiability of CTL* with constraints. In *CONCUR'13*, LNCS, pages 455–463. Springer, 2013.
15. C. Carapelle, A. Kartzow, and M. Lohrey. Satisfiability of ECTL* with constraints. *Journal of Computer and System Sciences*, 82(5):826–855, 2016.
16. C. Carapelle and A.-Y. Turhan. Description Logics Reasoning w.r.t. General TBoxes is Decidable for Concrete Domains with the EHD-property. In *ECAI'16*, volume 285, pages 1440–1448. IOS Press, 2016.
17. K. Čerāns. Deciding properties of integral relational automata. In *ICALP'94*, volume 820 of *LNCS*, pages 35–46. Springer, 1994.
18. A. Chandra, D. Kozen, and L. Stockmeyer. Alternation. *JACM*, 28(1):114–133, 1981.

19 R. Condurache, C. Dima, Y. Oualhdaj, and N. Troquard. Rational Synthesis in the Commons with Careless and Careful Agents. In *AAMAS'21*, pages 368–376, 2021.

20 B. Cook, H. Khlaaf, and N. Piterman. On automation of CTL* verification for infinite-state systems. In *CAV'15*, volume 9206 of *LNCS*, pages 13–29. Springer, 2015.

21 N. Decker, P. Habermehl, M. Leucker, and D. Thoma. Ordered navigation on multi-attributed data words. In *CONCUR'14*, volume 8704 of *LNCS*, pages 497–511, 2014.

22 S. Demri and M. Deters. Temporal logics on strings with prefix relation. *Journal of Logic and Computation*, 26:989–1017, 2016.

23 S. Demri and D. D'Souza. An automata-theoretic approach to constraint LTL. *I & C*, 205(3):380–415, 2007.

24 S. Demri and R. Gascon. Verification of qualitative $\mathbb{Z}$ constraints. *TCS*, 409(1):24–40, 2008.

25 S. Demri and R. Lazić. LTL with the freeze quantifier and register automata. *ACM ToCL*, 10(3), 2009.

26 S. Demri and K. Quaas. Concrete domains in logics: a survey. *ACM SIGLOG News*, 8(3):6–29, 2021.

27 S. Demri and K. Quaas. Constraint automata on infinite data trees: From CTL(Z)/CTL*(Z) to decision procedures. CoRR, abs/2302.05327, 2023.

28 A. Deutsch, R. Hull, and V. Vianu. Automatic verification of database-centric system. *SIGMOD Record*, 43(3):5–17, 2014.

29 E.A. Emerson and J. Halpern. 'sometimes' and 'not never' revisited: on branching versus linear time temporal logic. *JACM*, 33:151–178, 1986.

30 E.A. Emerson and C.S. Jutla. The complexity of tree automata and logics of programs. *SIAM Journal of Computing*, 29(1):132–158, 2000.

31 E.A. Emerson and P. Sistla. Deciding full branching time logic. *Information and Control*, 61:175–201, 1984.

32 L. Exibard, E. Filiot, and A. Khalimov. Church synthesis on register automata over linearly ordered data domains. In *STACS'21*, volume 187 of *LIPIcs*, pages 28:1–28:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

33 L. Exibard, E. Filiot, and A. Khalimov. A generic solution to register-bounded synthesis with an application to discrete orders. In *ICALP'22*, volume 229 of *LIPIcs*, pages 122:1–122:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

34 P. Felli, M. Montali, and S. Winkler. CTL* model checking for data-aware dynamic systems with arithmetic. In *IJCAR'22*, volume 13385 of *LNCS*, pages 36–56. Springer, 2022.

35 P. Felli, M. Montali, and S. Winkler. Linear-time verification of data-aware dynamic systems with arithmetic. In *AAAI'22*, pages 5642–5650. AAAI Press, 2022.

36 D. Figueira. *Reasoning on words and trees with data.* PhD thesis, ENS Cachan, 2010.

37 D. Figueira. Decidability of Downward XPath. *ACM ToCL*, 13(4):1–40, 2012.

38 R. Gascon. An automata-based approach for CTL* with constraints. *Electronic Notes in Theoretical Computer Science*, 239:193–211, 2009.

39 S. Göller, Ch. Haase, J. Ouaknine, and J. Worrell. Branching-time model checking of parametric one-counter automata. In *FoSSaCS'12*, volume 7213 of *LNCS*, pages 406–420. Springer, 2012.

40 J.F. Groote and R. Mastescu. Verification of temporal properties of processes in a setting with data. In *AMAST'98*, volume 1548 of *LNCS*, pages 74–90, 1998.

41 R. Iosif and X. Xu. Alternating automata modulo first order theories. In *CAV'19*, volume 11562 of *LNCS*, pages 46–63, 2019.

42 M. Jurdziński and R. Lazić. Alternating automata on data trees and XPath satisfiability. *ACM ToCL*, 12(3):19:1–19:21, 2011.

43 A. Kartzow and Th. Weidner. Model checking constraint LTL over trees. *CoRR*, abs/1504.06105, 2015.

44 O. Kupferman, M. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *JACM*, 47(2):312–360, 2000.

45 N. Labai. *Automata-based reasoning for decidable logics with data values*. PhD thesis, TU Wien, May 2021.

46 N. Labai, M. Ortiz, and M. Simkus. An Exptime Upper Bound for $\mathcal{ALC}$ with integers. In *KR'20*, pages 425–436. Morgan Kaufman, 2020.

47 S. Lasota and I. Walukiewicz. Alternating timed automata. *ACM ToCL*, 9(2):10:1–10:27, 2008.

48 A. Lechner, R. Mayr, J. Ouaknine, A. Pouly, and J. Worrell. Model checking flat freeze LTL on one-counter automata. *Logical Methods in Computer Science*, 14(4), 2018.

49 C. Lutz. Interval-based temporal reasoning with general TBoxes. In *IJCAI'01*, pages 89–94. Morgan-Kaufmann, 2001.

50 C. Lutz. *The Complexity of Description Logics with Concrete Domains*. PhD thesis, RWTH, Aachen, 2002.

51 C. Lutz. Description logics with concrete domains—a survey. In *Advances in Modal Logics Volume 4*, pages 265–296. King's College Publications, 2003.

52 C. Lutz. NEXPTIME-complete description logics with concrete domains. *ACM ToCL*, 5(4):669–705, 2004.

53 C. Lutz and M. Milicić. A Tableau Algorithm for Description Logics with Concrete Domains and General Tboxes. *JAR*, 38(1-3):227–259, 2007.

54 R. Mayr and P. Totzke. Branching-time model checking gap-order constraint systems. *Fundamenta Informaticae*, 143(3–4):339–353, 2016.

55 D. Muller and E. Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of Rabin, McNaughton and Safra. *TCS*, 141(1–2):69–107, 1995.

56 F. Neven, T. Schwentick, and V. Vianu. Finite state machines for strings over infinite alphabets. *ACM ToCL*, 5(3):403–435, 2004.

57 D. Peteler and K. Quaas. Deciding Emptiness for Constraint Automata on Strings with the Prefix and Suffix Order. In *MFCS'22*, volume 241 of *LIPIcs*, pages 76:1–76:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

58 M.O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the AMS*, 141:1–35, 1969.

59 P. Revesz. *Introduction to Constraint Databases*. Springer, New York, 2002.

60 S. Safra. *Complexity of Automata on Infinite Objects*. PhD thesis, The Weizmann Institute of Science, Rehovot, 1989.

61 L. Segoufin and S. Toruńczyk. Automata based verification over linearly ordered data domains. In *STACS'11*, pages 81–92, 2011.

62 H. Seidl, Th. Schwentick, and A. Muscholl. Counting in trees. In *Logic and Automata: History and Perspectives*, volume 2 of *Texts in Logic and Games*, pages 575–612. Amsterdam University Press, 2008.

63 F. Song and Z. Wu. On temporal logics with data variable quantifications: decidability and complexity. *I & C*, 251:104–139, 2016.

64 W. Thomas. Automata on infinite objects. In *Handbook of Theoretical Computer Science, Volume B, Formal models and semantics*, pages 133–191. Elsevier, 1990.

65 Sz. Torunczyk and Th. Zeume. Register automata with extrema constraints, and an application to two-variable logic. *Logical Methods in Computer Science*, 18(1), 2022.

66 M. Vardi and L. Stockmeyer. Improved upper and lower bounds for modal logics of programs. In *STOC'85*, pages 240–251. ACM, 1985.

67 M. Vardi and Th. Wilke. Automata: from logics to algorithms. In *Logic and Automata: History and Perspectives*, number 2 in Texts in Logic and Games, pages 629–736. Amsterdam University Press, 2008.

68 M. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Sciences*, 32:183–221, 1986.

69 M. Vardi and P. Wolper. Reasoning about infinite computations. *I & C*, 115:1–37, 1994.

762 **70** S. Vester. On the complexity of model-checking branching and alternating-time temporal
763 logics in one-counter systems. In *ATVA'15*, volume 9364 of *LNCS*, pages 361–377. Springer,
764 2015.
765 **71** Th. Weidner. *Probabilistic Logic, Probabilistic Regular Expressions, and Constraint Temporal*
766 *Logic*. PhD thesis, University of Leipzig, 2016.