



HAL
open science

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

Stéphane Demri, Karin Quaas

► **To cite this version:**

Stéphane Demri, Karin Quaas. Constraint Automata on Infinite Data Trees: from CTL()/ CTL*() to Decision Procedures. 34th International Conference on Concurrency Theory (CONCUR 2023), Sep 2023, Antwerp, Belgium. 10.4230/LIPIcs.CONCUR.2023.29 . hal-04201369

HAL Id: hal-04201369

<https://hal.science/hal-04201369>

Submitted on 28 Sep 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

1 Constraint Automata on Infinite Data Trees: 2 From $\text{CTL}(\mathbb{Z})/\text{CTL}^*(\mathbb{Z})$ To Decision Procedures

3 Stéphane Demri

4 Université Paris-Saclay, LMF, CNRS, ENS Paris-Saclay, France

5 Karin Quaas¹

6 Universität Leipzig, Fakultät für Mathematik und Informatik, Germany

7 — Abstract —

8 We introduce the class of tree constraint automata with data values in \mathbb{Z} (equipped with the less
9 than relation and equality predicates to constants), and we show that the nonemptiness problem is
10 EXPTIME-complete. Using an automata-based approach, we establish that the satisfiability problem
11 for $\text{CTL}(\mathbb{Z})$ (CTL with constraints in \mathbb{Z}) is EXPTIME-complete, and the satisfiability problem for
12 $\text{CTL}^*(\mathbb{Z})$ is 2EXPTIME-complete (only decidability was known so far). By-product results with
13 other concrete domains and other logics, are also briefly discussed.

14 **2012 ACM Subject Classification** Theory of computation \rightarrow Logic and verification

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

16 **Digital Object Identifier** 10.4230/LIPIcs.CONCUR.2023.29

17 **1** Introduction

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

30 Decidability results for concrete domains handled in [53, 38, 3] exclude the ubiquitous
31 concrete domain $(\mathbb{Z}, <, =, (=_{\mathfrak{d}})_{\mathfrak{d} \in \mathbb{Z}})$. By contrast, decidability results for logics with concrete
32 domain \mathbb{Z} require dedicated proof techniques, see e.g. [11, 23, 61, 46]. In particular, *fragments*
33 of $\text{CTL}^*(\mathbb{Z})$ are shown decidable in [11] using integral relational automata from [17], and
34 the satisfiability problem for existential and universal CTL^* with gap-order constraints
35 (more general than the ones in this paper) can be solved in PSPACE [12, Theorem 14].
36 Another important breakthrough came with the decidability of $\text{CTL}^*(\mathbb{Z})$ [15, Theorem 32]
37 (see also [14]) by designing a reduction to a decidable second-order logic, whose formulae are
38 made of Boolean combinations of formulae from MSO and from WMSO+U [10], where U is
39 the unbounding second-order quantifier, see e.g. [8, 9]. This is all the more remarkable as
40 the decidability result is part of a powerful general approach [15], but no sharp complexity
41 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 $\mathcal{ALCF}^P(\mathbb{Z}_c)$. 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}^β for some $\beta \geq 1$ (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/Streest 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 $\mathcal{ALCF}^P(\mathbb{Z}_c)$ 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

88 relations on \mathbb{Z} . Let $\text{VAR} = \{x, y, \dots\}$ be a countably infinite set of variables. A *term* \mathbf{t}
 89 *over* VAR is an expression of the form $X^i \mathbf{x}$, where $\mathbf{x} \in \text{VAR}$ and X^i is a (possibly empty)
 90 sequence of i symbols ‘ X ’. A term $X^i \mathbf{x}$ should be understood as a variable (that needs to be
 91 interpreted) but, later on, we will see that the prefix X^i will have a temporal interpretation.
 92 We write T_{VAR} to denote the set of all terms over VAR . For all $i \in \mathbb{N}$, we write $T_{\text{VAR}}^{\leq i}$ to
 93 denote the subset of terms of the form $X^j \mathbf{x}$, where $j \leq i$. For instance, $T_{\text{VAR}}^{\leq 0} = \text{VAR}$. An
 94 *atomic constraint* θ *over* T_{VAR} is an expression of one of the forms below:

$$95 \quad \mathbf{t} < \mathbf{t}' \quad \mathbf{t} = \mathbf{t}' \quad =_{\mathfrak{d}}(\mathbf{t}) \text{ (also written } \mathbf{t} = \mathfrak{d}\text{)},$$

96 where $\mathfrak{d} \in \mathbb{Z}$ and $\mathbf{t}, \mathbf{t}' \in T_{\text{VAR}}$. A *constraint* Θ is defined as a Boolean combination of
 97 atomic constraints. Constraints are interpreted on valuations $\mathbf{v} : T_{\text{VAR}} \rightarrow \mathbb{Z}$ that assign
 98 elements from \mathbb{Z} to the terms in T_{VAR} , so that \mathbf{v} *satisfies* θ , written $\mathbf{v} \models \theta$, if and only if, the
 99 interpretation of the terms in θ makes θ true in \mathbb{Z} in the usual way. The Boolean connectives
 100 are interpreted as usual. A constraint Θ is *satisfiable* $\stackrel{\text{def}}{\iff}$ there is a valuation $\mathbf{v} : T_{\text{VAR}} \rightarrow \mathbb{Z}$
 101 such that $\mathbf{v} \models \Theta$. Similarly, a constraint Θ_1 *entails* a constraint Θ_2 (written $\Theta_1 \models \Theta_2$) $\stackrel{\text{def}}{\iff}$
 102 for all valuations \mathbf{v} , we have $\mathbf{v} \models \Theta_1$ implies $\mathbf{v} \models \Theta_2$. The satisfiability problem restricted
 103 to finite conjunctions of atomic constraints can be solved in PTIME (see e.g. [17, Lemma
 104 5.5]) and entailment is in coNP. In the sequel, quite often, the valuations \mathbf{v} are of the form
 105 $\{\mathbf{x}_1, \dots, \mathbf{x}_\beta\} \rightarrow \mathbb{Z}$ when we are only interested in the values for the variables in $\{\mathbf{x}_1, \dots, \mathbf{x}_\beta\}$.

106 **Kripke structures.** In order to define logics with the concrete domain \mathbb{Z} , the semantical
 107 structures of such logics are enriched with valuations that interpret the variables by elements
 108 in \mathbb{Z} . A \mathbb{Z} -*decorated Kripke structure* (or *Kripke structure* for short) \mathcal{K} is a triple $(\mathcal{W}, \mathcal{R}, \mathbf{v})$,
 109 where \mathcal{W} is a non-empty set of *worlds*, $\mathcal{R} \subseteq \mathcal{W} \times \mathcal{W}$ is the accessibility relation and
 110 $\mathbf{v} : \mathcal{W} \times \text{VAR} \rightarrow \mathbb{Z}$ is a valuation function. A Kripke structure \mathcal{K} is *total* whenever for all
 111 $w \in \mathcal{W}$, there is $w' \in \mathcal{W}$ such that $(w, w') \in \mathcal{R}$. Given a Kripke structure $\mathcal{K} = (\mathcal{W}, \mathcal{R}, \mathbf{v})$
 112 and a world $w \in \mathcal{W}$, an *infinite path* π from w is an ω -sequence $w_0, w_1 \dots w_n, \dots$ such that
 113 $w_0 = w$ and for all $i \in \mathbb{N}$, we have $(w_i, w_{i+1}) \in \mathcal{R}$. Finite paths are defined accordingly.

114 **Labelled trees.** Given $D \geq 1$, a *labelled tree of degree D* is a map $\mathfrak{t} : \text{dom}(\mathfrak{t}) \rightarrow \Sigma$
 115 where Σ is some (potentially infinite) alphabet and $\text{dom}(\mathfrak{t})$ is an infinite subset of $[0, D-1]^*$
 116 such that $\mathbf{n} \in \text{dom}(\mathfrak{t})$ and $\mathbf{n} \cdot i \in \text{dom}(\mathfrak{t})$ for all $0 \leq i < j$ whenever $\mathbf{n} \cdot j \in \text{dom}(\mathfrak{t})$ for some
 117 $\mathbf{n} \in [0, D-1]^*$ and $j \in [0, D-1]$. The elements of $\text{dom}(\mathfrak{t})$ are called *nodes*. The empty
 118 word ε is the *root node* of \mathfrak{t} . For every $\mathbf{n} \in \text{dom}(\mathfrak{t})$, the elements $\mathbf{n} \cdot i$ ($i \in [0, D-1]$) are
 119 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
 120 \mathfrak{t} is a *full D -ary tree* if every node \mathbf{n} has exactly D children $\mathbf{n} \cdot 0, \dots, \mathbf{n} \cdot (D-1)$. Given a
 121 tree \mathfrak{t} and a node \mathbf{n} in $\text{dom}(\mathfrak{t})$, an infinite *path* in \mathfrak{t} starting from \mathbf{n} is an infinite sequence
 122 $\mathbf{n} \cdot j_1 \cdot j_2 \cdot j_3 \dots$, where $j_i \in [0, D-1]$ and $\mathbf{n} \cdot j_1 \dots j_i \in \text{dom}(\mathfrak{t})$ for all $i \geq 1$.

123 A *tree Kripke structure* \mathcal{K} is a Kripke structure $(\mathcal{W}, \mathcal{R}, \mathbf{v})$ such that $(\mathcal{W}, \mathcal{R})$ is a tree
 124 (not necessarily a full D -ary tree). Tree Kripke structures $(\mathcal{W}, \mathcal{R}, \mathbf{v})$ such that $(\mathcal{W}, \mathcal{R})$
 125 is isomorphic to the tree induced by $[0, D-1]^*$ are represented by maps of the form
 126 $\mathfrak{t} : [0, D-1]^* \rightarrow \mathbb{Z}^\beta$. This assumes that we only care about the value of the variables
 127 $\mathbf{x}_1, \dots, \mathbf{x}_\beta$ and $\mathfrak{t}(\mathbf{n}) = (\mathfrak{d}_1, \dots, \mathfrak{d}_\beta)$ encodes that for all $i \in [1, \beta]$, we have $\mathbf{v}(\mathbf{n}, \mathbf{x}_i) = \mathfrak{d}_i$.

128 2.2 The Logic CTL*(\mathbb{Z})

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

$$131 \quad \phi := \neg\phi \mid \phi \wedge \phi \mid \mathbf{E}\Phi \quad \Phi := \phi \mid \mathbf{t} = \mathfrak{d} \mid \mathbf{t}_1 = \mathbf{t}_2 \mid \mathbf{t}_1 < \mathbf{t}_2 \mid \neg\Phi \mid \Phi \wedge \Phi \mid X\Phi \mid \Phi U\Phi,$$

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

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

- 142 ■ $\mathcal{K}, w \models E\Phi \stackrel{\text{def}}{\iff}$ there is an infinite path π from w such that $\mathcal{K}, \pi \models \Phi$.
- 143 Let $\pi = w_0, w_1, \dots$ be an infinite path of \mathcal{K} . Let us define $\mathbf{v}(\pi, X^j \mathbf{x}) \stackrel{\text{def}}{=} \mathbf{v}(w_j, \mathbf{x})$, for all terms
 144 of the form $X^j \mathbf{x}$. For every n , $\pi[n, +\infty)$ is the suffix of π truncated by the n first worlds.
- 145 ■ $\mathcal{K}, \pi \models \mathbf{t} = \mathfrak{d} \stackrel{\text{def}}{\iff} \mathbf{v}(\pi, \mathbf{t}) = \mathfrak{d}$; $\mathcal{K}, \pi \models \mathbf{t}_1 \sim \mathbf{t}_2 \stackrel{\text{def}}{\iff} \mathbf{v}(\pi, \mathbf{t}_1) \sim \mathbf{v}(\pi, \mathbf{t}_2)$ for all $\sim \in \{<, =\}$,
- 146 ■ $\mathcal{K}, \pi \models \Phi U \Psi \stackrel{\text{def}}{\iff}$ there is $j \geq 0$ such that $\mathcal{K}, \pi[j, +\infty) \models \Psi$ and for all $j' \in [0, j - 1]$, we
 147 have $\mathcal{K}, \pi[j', +\infty) \models \Phi$;
- 148 ■ $\mathcal{K}, \pi \models X\Phi \stackrel{\text{def}}{\iff} \mathcal{K}, \pi[1, +\infty) \models \Phi$.

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

$$150 \quad \phi := E \Theta \mid A \Theta \mid \neg \phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid EX\phi \mid E\phi U\phi \mid E\phi R\phi \mid AX\phi \mid A\phi U\phi \mid A\phi R\phi,$$

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

156 The *satisfiability problem* for $\text{CTL}^*(\mathbb{Z})$, written $\text{SAT}(\text{CTL}^*(\mathbb{Z}))$, is defined as follows.

157 **Input:** A $\text{CTL}^*(\mathbb{Z})$ state formula ϕ .

158 **Question:** Is there a total Kripke structure \mathcal{K} and a world w such that $\mathcal{K}, w \models \phi$?

159 The satisfiability problem $\text{SAT}(\text{CTL}(\mathbb{Z}))$ for $\text{CTL}(\mathbb{Z})$ is defined analogously; for $\text{LTL}(\mathbb{Z})$,
 160 $\text{SAT}(\text{LTL}(\mathbb{Z}))$ is the problem to decide whether there exists an infinite sequence of valuations
 161 $\mathbf{v} : \text{VAR} \rightarrow \mathbb{Z}$ for a given $\text{LTL}(\mathbb{Z})$ formula Φ .

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

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

167 The proof in [14, 15] does not provide a complexity upper bound as the target decidable
 168 2nd-order logic admits an automata-based decision procedure with open complexity [10, 8, 9].

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

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

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

186 In this paper, we prove the precise computational complexity of $\text{SAT}(\text{CTL}^*(\mathbb{Z}))$ and
 187 $\text{SAT}(\text{CTL}(\mathbb{Z}))$. We follow the automata-based approach, that is, we translate formulae in
 188 our logics into equivalent automata – tree constraint automata for $\text{CTL}(\mathbb{Z})$, and Rabin tree
 189 constraint automata for $\text{CTL}^*(\mathbb{Z})$ – so that we can reduce the satisfiability problem for the
 190 logics to the nonemptiness problem for the corresponding automata.

191 3 Tree Constraint Automata

192 In this section, we introduce the class of tree constraint automata that accept sets of infinite
 193 trees of the form $\mathbb{t} : [0, D - 1]^* \rightarrow (\Sigma \times \mathbb{Z}^\beta)$ for some finite alphabet Σ and some $\beta \geq 1$.
 194 The transition relation of such automata states constraints between the β integer values
 195 at a node and the integer values at its children nodes. The acceptance condition is a
 196 Büchi condition (applied to the infinite branches of the input tree), but this can be easily
 197 extended to more general conditions (which we already consider by the end of this section).
 198 Moreover, our definition is specific to the concrete domain \mathbb{Z} but it can be easily adapted to
 199 other concrete domains. Formally, a *tree constraint automaton* (TCA, for short) is a tuple
 200 $\mathbb{A} = (Q, \Sigma, D, \beta, Q_{\text{in}}, \delta, F)$, where

- 201 ■ Q is a finite set of locations; Σ is a finite alphabet,
- 202 ■ $D \geq 1$ is the (branching) degree of (the trees accepted by) \mathbb{A} ,
- 203 ■ $\beta \geq 1$ is the number of variables (a.k.a. registers),
- 204 ■ $Q_{\text{in}} \subseteq Q$ is the set of initial locations; $F \subseteq Q$ encodes the Büchi acceptance condition,
- 205 ■ δ is a *finite* subset of $Q \times \Sigma \times (\text{TreeCons}(\beta) \times Q)^D$, the transition relation. Here,
 206 $\text{TreeCons}(\beta)$ denotes the constraints (Boolean combinations of atomic constraints) built
 207 over the terms $\mathbf{x}_1, \dots, \mathbf{x}_\beta, \mathbf{x}'_1, \dots, \mathbf{x}'_\beta$, where \mathbf{x}'_i denotes the term $\mathbb{X}\mathbf{x}_i$. δ consists of
 208 tuples $(q, \mathbf{a}, (\Theta_0, q_0), \dots, (\Theta_{D-1}, q_{D-1}))$, where $q \in Q$ is called the source location, $q_0, \dots,$
 209 $q_{D-1} \in Q$, $\mathbf{a} \in \Sigma$, and $\Theta_0, \dots, \Theta_{D-1}$ are constraints.

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

- 212 ■ $\rho(\varepsilon) = (q_{\text{in}}, \dots)$ such that $q_{\text{in}} \in Q_{\text{in}}$;
- 213 ■ for every $\mathbf{n} \in [0, D - 1]^*$ with $\rho(\mathbf{n}) = (q, \mathbf{a}, (\Theta_0, q_0), \dots, (\Theta_{D-1}, q_{D-1}))$, $\mathbb{t}(\mathbf{n} \cdot i) = (\mathbf{a}_i, \mathbf{z}_i)$,
 214 and $\rho(\mathbf{n} \cdot i)$ starts by the location q_i for all $0 \leq i < D$, we have $\mathbb{t}(\mathbf{n})$ of the form
 215 (\mathbf{a}, \mathbf{z}) and $\mathbb{Z} \models \Theta_i(\mathbf{z}, \mathbf{z}_i)$ for all $0 \leq i < D$. Here, $\mathbb{Z} \models \Theta_i(\mathbf{z}, \mathbf{z}_i)$ is a shortcut for
 216 $[\vec{\mathbf{x}} \leftarrow \mathbf{z}, \vec{\mathbf{x}}' \leftarrow \mathbf{z}_i] \models \Theta_i$ where $[\vec{\mathbf{x}} \leftarrow \mathbf{z}, \vec{\mathbf{x}}' \leftarrow \mathbf{z}_i]$ is a valuation \mathbf{v} on the variables
 217 $\{\mathbf{x}_j, \mathbf{x}'_j \mid j \in [1, \beta]\}$ with $\mathbf{v}(\mathbf{x}_j) = \mathbf{z}(j)$ and $\mathbf{v}(\mathbf{x}'_j) = \mathbf{z}_i(j)$ for all $j \in [1, \beta]$.

218 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
 219 $\pi = j_1 \cdot j_2 \cdot j_3 \dots$ in ρ starting from ε , we define $\text{inf}(\rho, \pi)$ to be the set of locations that appear
 220 infinitely often as the source locations of the transitions in $\rho(\varepsilon)\rho(j_1)\rho(j_1 \cdot j_2)\rho(j_1 \cdot j_2 \cdot j_3) \dots$.
 221 A run ρ is *accepting* if for all paths π in ρ starting from ε , we have $\text{inf}(\rho, \pi) \cap F \neq \emptyset$. We
 222 write $L(\mathbb{A})$ to denote the set of trees \mathbb{t} that admit an accepting run.

223 **Nonemptiness problem.** As usual, the *nonemptiness problem for TCA* asks whether
 224 a TCA \mathbb{A} satisfies $L(\mathbb{A}) \neq \emptyset$. To define the size of \mathbb{A} in a reasonably succinct encoding, we
 225 need to consider the size of constraints from $\text{TreeCons}(\beta)$. Indeed, unlike (plain) Büchi tree
 226 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 $\text{card}(\delta)$ is a priori unbounded, even if Q and Σ are fixed. We write $\text{MCS}(\mathbb{A})$ 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 $\mathbb{A} = (Q, \Sigma, D, \beta, Q_{\text{in}}, \delta, \mathcal{F})$ 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 $\rho : [0, D - 1]^* \rightarrow \delta$ is *accepting* iff for all paths π in ρ starting from ε , there is some $(L, U) \in \mathcal{F}$ such that $\text{inf}(\rho, \pi) \cap L \neq \emptyset$ and $\text{inf}(\rho, \pi) \cap U = \emptyset$.

Finite alphabet. The set Σ in data trees $\mathbb{t} : [0, D - 1]^* \rightarrow (\Sigma \times \mathbb{Z}^\beta)$ plays no specific role herein, especially that it could be encoded with simple constraints of the form $\mathbf{x}^* = \mathfrak{d}$, where \mathbf{x}^* 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 $[1, \text{card}(\Sigma)]$, 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, \emptyset) .

The proof of the EXPTIME upper bound is divided into two parts. In order to determine whether $L(\mathbb{A})$ is nonempty for a given TCA \mathbb{A} , we first reduce the existence of some tree $\mathbb{t} \in L(\mathbb{A})$ 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 $\mathbb{A} = (Q, \Sigma, D, \beta, Q_{\text{in}}, \delta, F)$ with the constants $\mathfrak{d}_1, \dots, \mathfrak{d}_\alpha$ occurring in \mathbb{A} such that $\mathfrak{d}_1 < \dots < \mathfrak{d}_\alpha$ (we assume there is at least one constant).

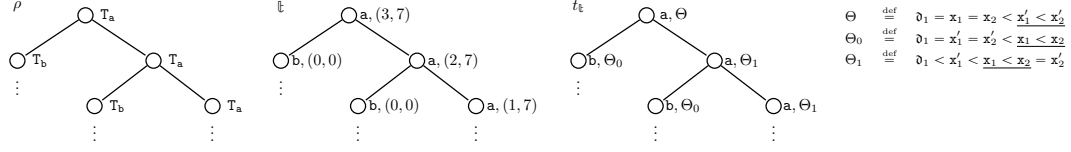
4.1 Symbolic Trees

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

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

■ for all $i \in [1, n]$, Θ_i^{CST} is equal to either $\mathbf{z}_i < \mathfrak{d}_1$, or $\mathbf{z}_i > \mathfrak{d}_\alpha$ or $\mathbf{z}_i = \mathfrak{d}$ 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 t (middle), a run ρ of some TCA on t (left), where, $T_a = (q, a, (\Theta_0, q), (\Theta_1, q))$ and $T_b = (q, b, (\Theta_0, q), (\Theta_1, q))$, and the symbolic tree $t_{\tilde{t}}$ (abstraction of t) (right).

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

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

274 Checking the satisfiability of a type can be done in polynomial-time, based on a standard
 275 cycle detection, see e.g. [17, Lemma 5.5]. The set of *satisfiable types* built over the variables
 276 $x_1, \dots, x_\beta, x'_1, \dots, x'_\beta$ is written $\text{SatTypes}(\beta)$ (n above is equal here to 2β). Observe that
 277 $\text{card}(\text{SatTypes}(\beta)) \leq ((\mathfrak{d}_\alpha - \mathfrak{d}_1) + 3)^{2\beta} \times 3^{\beta^2}$. The *restriction* of the type Θ to some set
 278 of variables $X \subseteq \{x_i, x'_i \mid i \in [1, \beta]\}$ is made of all the conjuncts in which only variables in
 279 X occur. The type Θ restricted to $\{x'_i \mid i \in [1, \beta]\}$ *agrees* with the type Θ' restricted to
 280 $\{x_i \mid i \in [1, \beta]\}$ iff Θ and Θ' are logically equivalent modulo the renaming for which x_i and x'_i
 281 are substituted, for all $i \in [1, \beta]$. For instance, in Figure 1, Θ restricted to $\{x'_1, x'_2\}$ agrees
 282 with Θ_0 restricted to $\{x_1, x_2\}$. The main properties we use about satisfiable types are stated
 283 below.

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

285 (II) For every constraint Θ built over the variables $x_1, \dots, x_\beta, x'_1, \dots, x'_\beta$ and the constants
 286 $\mathfrak{d}_1, \dots, \mathfrak{d}_\alpha$ there is a disjunction $\Theta_1 \vee \dots \vee \Theta_\gamma$ logically equivalent to Θ and each Θ_i
 287 belongs to $\text{SatTypes}(\beta)$ (empty disjunction stands for \perp).

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

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

290 **Abstraction with types.** A *symbolic tree* t is a map $t : [0, D - 1]^* \rightarrow \Sigma \times \text{SatTypes}(\beta)$.
 291 Symbolic trees are intended to be abstractions of trees labelled with concrete values in \mathbb{Z} .
 292 Given a tree $\mathfrak{t} : [0, D - 1]^* \rightarrow \Sigma \times \mathbb{Z}^\beta$, its *abstraction* is the symbolic tree $t_{\tilde{t}} : [0, D - 1]^* \rightarrow$
 293 $\Sigma \times \text{SatTypes}(\beta)$ such that for all $\mathbf{n} \cdot i \in [0, D - 1]^*$ with $\mathfrak{t}(\mathbf{n}) = (\mathbf{a}, \mathbf{z})$ and $\mathfrak{t}(\mathbf{n} \cdot i) = (\mathbf{a}_i, \mathbf{z}_i)$,
 294 $t_{\tilde{t}}(\mathbf{n} \cdot i) \stackrel{\text{def}}{=} (\mathbf{a}_i, \Theta_i)$ for the unique $\Theta_i \in \text{SatTypes}(\beta)$ such that $\mathbb{Z} \models \Theta_i(\mathbf{z}, \mathbf{z}_i)$. Note that
 295 the primed variables in Θ_i refer to the β values at the node $\mathbf{n} \cdot i$, whereas the unprimed
 296 ones refer to the β values at the parent node \mathbf{n} . At the root ε with $\mathfrak{t}(\varepsilon) = (\mathbf{a}, \mathbf{z})$, we have
 297 $t_{\tilde{t}}(\varepsilon) \stackrel{\text{def}}{=} (\mathbf{a}, \Theta)$ for the unique $\Theta \in \text{SatTypes}(\beta)$ such that $\mathbb{Z} \models \Theta(\mathbf{0}, \mathbf{z})$, where $\mathbf{0} \in \mathbb{Z}^\beta$ is
 298 arbitrary as there are actually no parent values at the root. A symbolic tree t is *satisfiable*
 299 $\stackrel{\text{def}}{\iff}$ there is $\mathfrak{t} : [0, D - 1]^* \rightarrow \Sigma \times \mathbb{Z}^\beta$ such that $t_{\tilde{t}} = t$. We say that \mathfrak{t} *witnesses the satisfaction*
 300 *of* t , also written $\mathfrak{t} \models t$. A symbolic tree t is *regular* if its set of subtrees is finite.

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

304 ■ t is *locally consistent*: for every node \mathbf{n} , the type Θ labelling \mathbf{n} restricted to x'_1, \dots, x'_β
 305 agrees with all types Θ_i labelling its children nodes $\mathbf{n} \cdot i$ restricted to x_1, \dots, x_β , and

306 ■ there is an accepting run ρ of \mathbb{A} (but ignoring the conditions on data values) such that
 307 for all $\mathbf{n} \in [0, D - 1]^*$ with $t(\mathbf{n}) = (\mathbf{a}, \Theta)$, $t(\mathbf{n} \cdot i) = (\mathbf{a}_i, \Theta_i)$ for all $i \in [0, D - 1]$, and
 308 $\rho(\mathbf{n}) = (q, \mathbf{a}, (\Theta'_0, q_0) \dots (\Theta'_{D-1}, q_{D-1}))$, we have $\Theta_i \models \Theta'_i$ for all $i \in [0, D - 1]$.

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

316 ▶ **Lemma 3.** *There exists a Büchi tree automaton (without constraints) $A_{\text{cons}(\mathbb{A})}$ such that*
 317 $L(A_{\text{cons}(\mathbb{A})})$ *is equal to the set of \mathbb{A} -consistent symbolic trees.*

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

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

325 ▶ **Lemma 4.** $L(\mathbb{A}) \neq \emptyset$ *iff there is a satisfiable symbolic tree in $L(A_{\text{cons}(\mathbb{A})})$.*

326 4.2 Satisfiability for Regular Locally Consistent Symbolic Trees

327 Below, we focus on deciding when $L(A_{\text{cons}(\mathbb{A})})$ contains a *satisfiable* symbolic tree, while
 328 evaluating the complexity to check its existence. Given a locally consistent symbolic tree
 329 $t : [0, D - 1]^* \rightarrow \Sigma \times \text{SatTypes}(\beta)$, we introduce an infinite labelled graph that contains
 330 exactly the same types as t but expressed in a tree-like graph from which it is convenient
 331 to characterize satisfiability in terms of paths, under the premise that t is regular. Similar
 332 symbolic structures are introduced in [49, 23, 14, 46]. The graph is equal to the structure

$$333 G_t^c = (V_t, \overset{=}{\rightarrow}, \overset{<}{\rightarrow}, U_{<\mathfrak{d}_1}, (U_i)_{i \in [\mathfrak{d}_1, \mathfrak{d}_\alpha]}, U_{>\mathfrak{d}_\alpha}),$$

334 where $V_t = [0, D - 1]^* \times (\{\mathbf{x}_1, \dots, \mathbf{x}_\beta\} \cup \{\mathfrak{d}_1, \mathfrak{d}_\alpha\})$, $\overset{=}{\rightarrow}$ and $\overset{<}{\rightarrow}$ are two binary relations over V_t ,
 335 and $\{U_{<\mathfrak{d}_1}, U_{\mathfrak{d}_1}, U_{\mathfrak{d}_1+1}, \dots, U_{\mathfrak{d}_\alpha}, U_{>\mathfrak{d}_\alpha}\}$ is a partition of V_t . Elements in $\{\mathbf{x}_1, \dots, \mathbf{x}_\beta\} \cup \{\mathfrak{d}_1, \mathfrak{d}_\alpha\}$
 336 are denoted by $\mathbf{x}, \mathbf{x}_1, \mathbf{x}_2, \dots$ (variables or constants). Moreover, $V_t^\beta \stackrel{\text{def}}{=} [0, D - 1]^* \times$
 337 $\{\mathbf{x}_1, \dots, \mathbf{x}_\beta\}$. The rationale behind the construction of G_t^c is to reflect the constraints between
 338 parent and children nodes as well as the constraints regarding constants, in such a way that,
 339 if \mathbb{t} witnesses the satisfaction of t , then, e.g., $\mathbb{t}(\mathbf{n})(\mathbf{x}, \mathbf{d}) < \mathbb{t}(\mathbf{n}')(\mathbf{x}', \mathbf{d}')$ if $(\mathbf{n}, \mathbf{x}, \mathbf{d}) \overset{<}{\rightarrow} (\mathbf{n}', \mathbf{x}', \mathbf{d}')$, and
 340 $\mathbb{t}(\mathbf{n})(\mathbf{x}, \mathbf{d}) = \mathfrak{d}_1$ if $(\mathbf{n}, \mathbf{x}, \mathbf{d}) \in U_{\mathfrak{d}_1}$. Here are all conditions for building G_t^c .

341 **(VAR)** For all $(\mathbf{n}, \mathbf{x}_i), (\mathbf{n}', \mathbf{x}_{i'}) \in V_t^\beta$, for all $\sim \in \{<, =\}$, $(\mathbf{n}, \mathbf{x}_i) \overset{\sim}{\rightarrow} (\mathbf{n}', \mathbf{x}_{i'})$ iff either $\mathbf{n}' = \mathbf{n} \cdot j$
 342 and $\mathbf{x}_i \sim \mathbf{x}_{i'}$ in Θ with $t(\mathbf{n}') = (\mathbf{a}, \Theta)$, or $\mathbf{n} = \mathbf{n}'$ and $\mathbf{x}'_i \sim \mathbf{x}'_{i'}$ in Θ with $t(\mathbf{n}') = (\mathbf{a}, \Theta)$, or
 343 $\mathbf{n} = \mathbf{n}' \cdot j$ and $\mathbf{x}'_i \sim \mathbf{x}_{i'}$ in Θ with $t(\mathbf{n}') = (\mathbf{a}, \Theta)$.

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

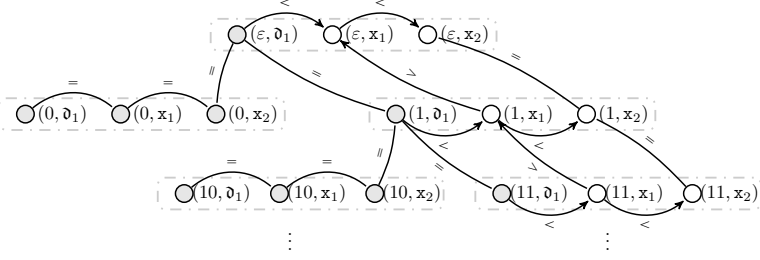
345 **(P2)** For all $(\mathbf{n}, \mathbf{x}_j) \in V_t^\beta$, $(\mathbf{n}, \mathbf{x}_j) \in U_{<\mathfrak{d}_1}$ iff $\mathbf{x}'_j < \mathfrak{d}_1$ in Θ with $t(\mathbf{n}) = (\mathbf{a}, \Theta)$.

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

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

348 **(CONS)** This is about elements of V_t labelled by constants and how the edge labels reflect the
 349 relationships between the constants. Formally, for all $((\mathbf{n}, \mathbf{x}, \mathbf{d}), (\mathbf{n}', \mathbf{x}', \mathbf{d}')) \in (V_t \times V_t) \setminus (V_t^\beta \times$
 350 $V_t^\beta)$, for all $\mathfrak{d}^\dagger, \mathfrak{d}^{\dagger\dagger}$ in $\{< \mathfrak{d}_1, \mathfrak{d}_1, \dots, \mathfrak{d}_\alpha, > \mathfrak{d}_\alpha\}$ s.t. $(\mathbf{n}, \mathbf{x}, \mathbf{d}) \in U_{\mathfrak{d}^\dagger}$ and $(\mathbf{n}', \mathbf{x}', \mathbf{d}') \in U_{\mathfrak{d}^{\dagger\dagger}}$, for
 351 all $\sim \in \{<, =\}$, $(\mathbf{n}, \mathbf{x}, \mathbf{d}) \overset{\sim}{\rightarrow} (\mathbf{n}', \mathbf{x}', \mathbf{d}')$ 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 $\mathfrak{d}^\dagger = \{< \mathfrak{d}_1\}$,
 352 $\mathfrak{d}^{\dagger\dagger} \neq \{< \mathfrak{d}_1\}$ 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 $<$.

353 Below, we illustrate the definition of the graph $G_{t_{\mathbb{k}}}^C$ for the symbolic tree $t_{\mathbb{k}}$ in Figure 1.
 354 The edges labelled with $=$ or $<$ reflect the constraints (we omit edges if they can be inferred
 from the other edges).
 For instance, $(1, \mathbf{x}_1) \overset{<}{\sim} (\varepsilon, \mathbf{x}_1)$ corresponds to the
 355 constraint $\mathbf{x}'_1 < \mathbf{x}_1$. Grey nodes are in $U_{\mathfrak{d}_1}$, all other
 nodes are in $U_{>\mathfrak{d}_1}$ (no nodes in $U_{<\mathfrak{d}_1}$).



356 A map $p : \mathbb{N} \rightarrow V_t$ is a *path map* in $G_t^C \stackrel{\text{def}}{\iff}$ for all $i \in \mathbb{N}$, either $p(i) \overset{=}{\rightarrow} p(i+1)$ or
 357 $p(i) \overset{<}{\sim} p(i+1)$ in G_t^C . Similarly, $r : \mathbb{N} \rightarrow V_t$ is a *reverse path map* in $G_t^C \stackrel{\text{def}}{\iff}$ for all $i \in \mathbb{N}$,
 358 either $r(i) \overset{=}{\rightarrow} r(i+1)$ or $r(i+1) \overset{<}{\sim} r(i)$. A path map p (resp. reverse path map r) is
 359 *strict* $\stackrel{\text{def}}{\iff} \{i \in \mathbb{N} \mid p(i) \overset{<}{\sim} p(i+1)\}$ (resp. $\{i \in \mathbb{N} \mid r(i+1) \overset{<}{\sim} r(i)\}$) is infinite. An *infinite*
 360 *branch* \mathcal{B} is an element of $[0, D-1]^\omega$. We write $\mathcal{B}[i, j]$ with $i \leq j$ to denote the subsequence
 361 $\mathcal{B}(i) \cdot \mathcal{B}(i+1) \cdots \mathcal{B}(j)$. Given $(\mathbf{n}, \mathbf{xd}) \in V_t$, a path map p from $(\mathbf{n}, \mathbf{xd})$ along \mathcal{B} is such that
 362 $p(0) = (\mathbf{n}, \mathbf{xd})$ and for all $i \geq 0$, $p(i)$ is of the form $(\mathbf{n} \cdot \mathcal{B}[0, i], \cdot)$. A reverse path map r from
 363 $(\mathbf{n}, \mathbf{xd})$ along \mathcal{B} admits a similar definition. We present the condition (\star^C) that is the central
 364 property for characterising regular symbolic trees in $L(A_{\text{cons}(\mathcal{A})})$ that are satisfiable, following
 365 the remarkable result established in [46, Lemma 22] that non-satisfiability of a symbolic tree
 366 can be witnessed along a *single branch*.

367 (\star^C) There are *no* elements $(\mathbf{n}, \mathbf{xd}), (\mathbf{n}, \mathbf{xd}')$ in G_t^C (same node \mathbf{n} from $[0, D-1]^*$) and no
 368 infinite branch \mathcal{B} such that
 369 1. there exists a path map p from $(\mathbf{n}, \mathbf{xd})$ along \mathcal{B} ,
 370 2. there exists a reverse path map r from $(\mathbf{n}, \mathbf{xd}')$ along \mathcal{B} ,
 371 3. p or r is strict, and
 372 4. for all $i \in \mathbb{N}$, $p(i) \overset{<}{\sim} r(i)$.

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

375 **► Proposition 5.** For every regular locally consistent symbolic tree t , G_t^C satisfies (\star^C) iff t is
 376 satisfiable.

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

378 **► Example 6.** Assume that every node along the rightmost branch in the symbolic tree
 379 $t_{\mathbb{k}}$ in Figure 1 is labelled with (\mathbf{a}, Θ_1) . Then $t_{\mathbb{k}}$ is not satisfiable: in order to satisfy Θ_1 's
 380 conjunct $\mathbf{x}'_1 < \mathbf{x}_1$, the value of \mathbf{x}_1 must inevitably become finally smaller than \mathfrak{d}_1 , violating
 381 the conjunct $\mathfrak{d}_1 < \mathbf{x}_1$. Consequently, the rightmost branch of $G_{t_{\mathbb{k}}}^C$ presented above does not
 382 satisfy (\star^C) : there exists a path map p from $(\varepsilon, \mathfrak{d}_1)$ along 1^ω , there exists a strict reverse
 383 path map r from $(\varepsilon, \mathbf{x}_1)$ along 1^ω , and for all $i \in \mathbb{N}$ we have $p(i) \overset{<}{\sim} r(i)$.

384 **New constant nodes.** Proposition 5 above is a variant of [46, Lemma 22]. Before going
 385 any further, let us in short explain the improvement of our developments compared to what is
 386 done in [46, 45]. The *framified constraint graphs* defined in [46, Definition 14] correspond to
 387 the above defined graph G_t^C without $[0, D-1]^* \times \{\mathfrak{d}_1, \mathfrak{d}_\alpha\}$ and corresponding edges. However,
 388 Example 6 illustrates the importance of taking into account these elements when deciding
 389 satisfiability (without \mathfrak{d}_1 , the graph would satisfy (\star^C)). Actually, Example 6 invalidates (\star)
 390 as used in [46, 45] because the constants are missing to apply properly [15]. The problematic
 391 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 (\star) in [46, Section 3.3] generalises the condition C_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 (\star) 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 (\star^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 (\star^c) is ω -regular; hence, satisfiability of symbolic trees can be overapproximated advantageously.

► **Lemma 7.** *There is a Rabin tree automaton A_{\star^c} such that $L(A_{\star^c}) = \{t \mid G_t^c \text{ satisfies } (\star^c)\}$, 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*

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

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 (\star^c) for $D = 1$. (2) A_B is nondeterministic, but we can determinize it and get a deterministic Rabin word automaton $A_{B \rightarrow R}$ such that $L(A_B) = L(A_{B \rightarrow R})$ (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 $L(A_{B \rightarrow R})$; it accepts words that satisfy (\star^c) for $D = 1$. (4) By [60, Lemma 1.2], we construct a deterministic Rabin word automaton A_R s.t. $L(A_S) = L(A_R)$. (5) Finally, we construct a Rabin tree automaton A_{\star^c} , the intuitive idea is to "let run the automaton A_R " along every branch of a run of A_{\star^c} , doable thanks to the determinism of A_R . Since (\star^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_{\star^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_{\star^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 (\star^c) , but also we control the size parameters of A_{\star^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.** $L(\mathbb{A}) \neq \emptyset$ iff $L(A_{\text{cons}(\mathbb{A})}) \cap L(A_{\star^c}) \neq \emptyset$.

For its proof, by way of example, if $L(A_{\text{cons}(\mathbb{A})}) \cap L(A_{\star^c})$ is non-empty, then as $L(A_{\text{cons}(\mathbb{A})}) \cap L(A_{\star^c})$ 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 $L(\mathbb{A}) \neq \emptyset$. 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 (\star^c) .

4.3 ExpTime Upper Bound for TCAs

Lemma 8 justifies why deciding the nonemptiness of $L(A_{\text{cons}(\mathbb{A})}) \cap L(A_{\star^c})$ 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 $L(A) = L(A_{\text{cons}(\mathbb{A})}) \cap L(A_{\star^c})$ and the number of Rabin pairs is polynomial in β , the number of locations is in $\mathcal{O}(\text{card}(\text{SatTypes}(\beta)) \times \text{card}(Q) \times 2^{P(\beta)})$ for some polynomial $P(\cdot)$ and the transition relation can be decided in polynomial-time in $\text{card}(\delta) + \beta + \text{card}(\Sigma) + D + \text{MCS}(\mathbb{A})$.*

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, $D \geq 1$ and deciding whether a tuple belongs to δ requires γ time units, checking nonemptiness is actually in time $(\text{card}(\delta) \times \gamma \times n)^{\mathcal{O}(n)}$ (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 $\text{card}(\delta) + \beta + \text{card}(\Sigma) + D + \text{MCS}(\mathbb{A})$ (by Lemma 9). Hence the following result:

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

Assuming that the size of the TCA $\mathbb{A} = (Q, \Sigma, D, \beta, Q_{\text{in}}, \delta, F)$, written $\text{size}(\mathbb{A})$, is polynomial in $\text{card}(Q) + \text{card}(\delta) + D + \beta + \text{MCS}(\mathbb{A})$ (which makes sense for a reasonably succinct encoding), from the computation of the bound in Lemma 10, the nonemptiness of $L(\mathbb{A})$ can be checked in time $\mathcal{O}(R(\text{size}(\mathbb{A}))^{R'(\beta+D)})$ 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 $\mathbb{A} = (Q, \Sigma, D, \beta, Q_{\text{in}}, \delta, \mathcal{F})$, we define a Rabin tree automaton $A'_{\text{cons}(\mathbb{A})}$ such that $L(\mathbb{A}) \neq \emptyset$ iff there is $t \in L(A'_{\text{cons}(\mathbb{A})})$ that is satisfiable (cf. Lemma 4 for TCA). Moreover, we take advantage of A_{\star^c} so that $L(\mathbb{A}) \neq \emptyset$ iff $L(A'_{\text{cons}(\mathbb{A})}) \cap L(A_{\star^c})$ is non-empty (cf. Lemma 8). It remains to determine the cost for testing nonemptiness of $L(A'_{\text{cons}(\mathbb{A})}) \cap L(A_{\star^c})$. Here is the counterpart of Lemma 9 (same kind of arguments).

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

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

480 ▶ **Theorem 13.** *The nonemptiness problem for Rabin TCA is EXPTIME-complete.*

481 This result is mainly useful to characterize the complexity of $\text{SAT}(\text{CTL}^*(\mathbb{Z}))$ in Section 6.

482 **5 Tree Constraint Automata for CTL(\mathbb{Z})**

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

488 ▶ **Theorem 14.** *The satisfiability problem for $\text{CTL}(\mathbb{Z})$ is EXPTIME-complete.*

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

- 493 ■ The degree D and the number of variables β are bounded by $\text{size}(\phi)$.
- 494 ■ The number of locations is bounded by $(D \times 2^{\text{size}(\phi)}) \times (\text{size}(\phi) + 1)$.
- 495 ■ The number of transitions is in $\mathcal{O}(2^{P(\text{size}(\phi))})$ for some polynomial $P(\cdot)$.
- 496 ■ The finite alphabet Σ in A_ϕ is unary; $\text{MCS}(A_\phi)$ is quadratic in $\text{size}(\phi)$.

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

$$498 \mathcal{O}(R_1(\text{card}(Q) \times \text{card}(\delta) \times \text{MCS}(A) \times \text{card}(\Sigma) \times R_2(\beta))^{R_2(\beta) \times R_3(D)}).$$

499 Since the transition relations of the automata $A_{\text{cons}(A)}$ and $A_{\star c}$ can be built in polynomial-time,
 500 we get that nonemptiness of $L(A_\phi)$ can be solved in exponential-time. ◀

501 Let \mathbb{N} be the concrete domain $(\mathbb{N}, <, =, (=_{\mathfrak{d}})_{\mathfrak{d} \in \mathbb{N}})$ for which we can also show that nonemptiness
 502 of TCA with constraints interpreted on \mathbb{N} has the same complexity as for TCA with constraints
 503 interpreted on \mathbb{Z} . Let $\text{CTL}(\mathbb{N})$ be the variant of $\text{CTL}(\mathbb{Z})$ with constraints interpreted
 504 on \mathbb{N} . As a corollary, $\text{SAT}(\text{CTL}(\mathbb{N}))$ is EXPTIME-complete. With the concrete domain
 505 $(\mathbb{Q}, <, =, (=_{\mathfrak{d}})_{\mathfrak{d} \in \mathbb{Q}})$, all the trees in $L(A_{\text{cons}(A)})$ are satisfiable (no need to intersect $A_{\text{cons}(A)}$ with
 506 a hypothetical $A_{\star c}$, see e.g. [49, 4, 23, 38]), and therefore $\text{SAT}(\text{CTL}(\mathbb{Q}))$ is in EXPTIME too.
 507 TCA can be also used to show that the concept satisfiability w.r.t. general TBoxes for the
 508 description logic $\mathcal{ALCF}^P(\mathbb{Z}_c)$ is in EXPTIME [46, 45], see more details in [27, Section 5.2].

509 **6 Complexity of the Satisfiability Problem for the Logic $\text{CTL}^*(\mathbb{Z})$**

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

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

$$517 \text{E} (x = 0) \wedge \left(\bigwedge_{i \in [1, D-1]} \text{AGE } \Phi_i \right) \wedge \left(\bigwedge_{j \in [1, D']} \text{A } \Phi'_j \right),$$

518 where the Φ_i 's and the Φ'_j 's are $\text{LTL}(\mathbb{Z})$ formulae in simple form (see Section 2), for some
 519 $D \geq 1, D' \geq 0$. We can restrict ourselves to $\text{CTL}^*(\mathbb{Z})$ state formulae in special form (see
 520 the proof of [27, Proposition 6]).

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

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

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

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

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

540 We can now construct, for every $i \in [0, D - 1]$, a TCA \mathbb{A}_i such that $\text{L}(\mathbb{A}_i) = \{\mathbb{t} : [0, D - 1]^* \rightarrow \mathbb{Z}^\beta \mid \mathbb{t} \models \text{AGE } \Phi_i \text{ and } \mathbb{t} \text{ satisfies AGE } \Phi_i \text{ via } i\}$. The idea is to construct \mathbb{A}_i so
 541 that it starts off the word constraint automaton \mathbb{A}_{Φ_i} at each node \mathbf{n} of the tree and runs it
 542 down the designated path $\mathbf{n} \cdot i \cdot 0^\omega$ to check whether Φ_i actually holds along this path. This
 543 can be easily done for AGE Φ_i ; however, for formulas of the form A Φ'_j , for this construction
 544 to be correct, the underlying constraint word automaton \mathbb{A}'_j must be *deterministic*, that
 545 is, for all locations s , letters \mathbf{a} and pairs of valuations $(\mathbf{z}, \mathbf{z}') \in \mathbb{Z}^{2\beta}$, there exists in \mathbb{A}'_j at
 546 most a single transition $(s, \mathbf{a}, \Theta, s')$ such that $\mathbb{Z} \models \Theta(\mathbf{z}, \mathbf{z}')$. A well-known construction to
 547 transform nondeterministic Büchi automata to equivalent deterministic Rabin automata
 548 is due to Safra [60, Theorem 1.1]. An important step towards the optimal complexity for
 549 $\text{CTL}^*(\mathbb{Z})$ is to show that it is possible to lift this construction to word constraint automata,
 550 which is a result of its own interest. A special attention is given to the cardinality of the
 551 transition relation and to the size of the constraints in transitions, as these two parameters
 552 are, a priori, unbounded in constraint automata but essential to perform a forthcoming
 553 complexity analysis.
 554

555 ▶ **Theorem 18.** *Let $\mathbb{A} = (Q, \Sigma, \beta, Q_{in}, \delta, F)$ be a Büchi word constraint automaton involving*
 556 *the constants $\mathfrak{d}_1, \dots, \mathfrak{d}_\alpha$. There is a deterministic Rabin word constraint automaton $\mathbb{A}' =$*
 557 *$(Q', \Sigma, \beta, Q'_{in}, \delta', \mathcal{F})$ such that $\text{L}(\mathbb{A}) = \text{L}(\mathbb{A}')$ verifying the following quantitative properties.*
 558 *(I) $\text{card}(Q')$ is exponential in $\text{card}(Q)$ and the number of Rabin pairs in \mathbb{A}' is bounded by*
 559 *$2 \cdot \text{card}(Q)$ (same bounds as in [60, Theorem 1.1]).*
 560 *(II) The constraints in the transitions are from $\text{SatTypes}(\beta)$, are of size cubic in $\beta +$
 561 *$\max(\lceil \log(|\mathfrak{d}_1|) \rceil, \lceil \log(|\mathfrak{d}_\alpha|) \rceil)$ and $\text{card}(\delta') \leq \text{card}(Q')^2 \times \text{card}(\Sigma) \times ((\mathfrak{d}_\alpha - \mathfrak{d}_1) + 3)^{2\beta} \times 3^{\beta^2}$.**

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

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

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

570 This enables us to use the idea illustrated above for formulas of the form AGE Φ_i also for
 571 formulas of the form A Φ'_j , and define Rabin TCA \mathbb{A}'_j such that $L(\mathbb{A}'_j) = \{\mathfrak{t} : [0, D-1]^* \rightarrow$
 572 $\mathbb{Z}^\beta \mid \mathfrak{t} \text{ satisfies A } \Phi'_j\}$. We are now ready to perform the final step towards the main result of
 573 this section. Let us recapitulate what we have so far.

- 574 ■ One can define a TCA \mathbb{A}_0 with two locations such that $L(\mathbb{A}_0)$ is the set of trees $\mathfrak{t} :$
 575 $[0, D-1]^* \rightarrow \mathbb{Z}^\beta$ such that $\mathfrak{t}(\varepsilon)(x_1) = 0$, to handle $E(x_1 = 0)$ in formulae in special form.
- 576 ■ For all $1 \leq i < D$, there are (Büchi) TCA \mathbb{A}_i such that $L(\mathbb{A}_i)$ is the set of trees
 577 $\mathfrak{t} : [0, D-1]^* \rightarrow \mathbb{Z}^\beta$ such that $\mathfrak{t}, \varepsilon \models \text{AGE } \Phi_i$ and \mathfrak{t} satisfies AGE Φ_i via i . Recall that
 578 TCA can be seen as Rabin TCA with a single Rabin pair.
- 579 ■ For all $1 \leq j \leq D'$, there are Rabin TCA \mathbb{A}'_j such that $L(\mathbb{A}'_j)$ is the set of trees \mathfrak{t} such
 580 that \mathfrak{t} satisfies A Φ'_j , with an exponential number of Rabin pairs in $\text{size}(\Phi)$.

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

584 ► **Lemma 20.** *Let $(\mathbb{A}_k)_{1 \leq k \leq n}$ be a family of Rabin TCA such that $\mathbb{A}_k = (Q_k, \Sigma, D, \beta, Q_{k, \text{in}}, \delta_k, \mathcal{F}_k)$,
 585 $\text{card}(\mathcal{F}_k) = N_k$ and $N = \prod_k N_k$. There is a Rabin TCA \mathbb{A} such that $L(\mathbb{A}) = \bigcap_k L(\mathbb{A}_k)$ and*

- 586 ■ *the number of Rabin pairs is equal to N ; $\text{MCS}(\mathbb{A}) \leq n + \text{MCS}(\mathbb{A}_1) + \dots + \text{MCS}(\mathbb{A}_n)$,*
- 587 ■ *the number of locations (resp. transitions) is less than $(\prod_k \text{card}(Q_k))(2n)^N$ (resp. $\prod_k \text{card}(\delta_k)$).*

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

591 ► **Theorem 21.** *SAT(CTL*(\mathbb{Z})) is 2EXPTIME-complete.*

592 2EXPTIME-hardness is from SAT(CTL*) [66, Theorem 5.2]. As a corollary, SAT(CTL*(\mathbb{N}))
 593 is also 2EXPTIME-complete. Furthermore, assuming that $<_{\text{pre}}$ is the prefix relation on $\{0, 1\}^*$,
 594 we can use the reduction from [22, Section 4.2] to conclude SAT(CTL*($\{0, 1\}^*, <_{\text{pre}}$)) is
 595 2EXPTIME-complete too. Furthermore, as observed earlier, when the concrete domain is
 596 $(\mathbb{Q}, <, =, (=_{\mathfrak{d}})_{\mathfrak{d} \in \mathbb{Q}})$, all the trees in $L(\mathbb{A}_{\text{cons}(\mathbb{A})})$ are satisfiable, and therefore SAT(CTL*(\mathbb{Q})) is
 597 also in 2EXPTIME, which is already known from [38, Theorem 4.3].

598 7 Concluding Remarks

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

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

614 This lead us to show that $\text{SAT}(\text{CTL}(\mathbb{Z}))$ is EXPTIME -complete (Theorem 14), and
 615 $\text{SAT}(\text{CTL}^*(\mathbb{Z}))$ is 2EXPTIME -complete (Theorem 21). The only decidability proof for
 616 $\text{SAT}(\text{CTL}^*(\mathbb{Z}))$ done so far, see [15, Theorem 32], is by reduction to a decidable second-order
 617 logic. Our complexity characterisation for $\text{SAT}(\text{CTL}^*(\mathbb{Z}))$ provides an answer to several
 618 open problems related to $\text{CTL}^*(\mathbb{Z})$ fragments, see e.g. [11, 38, 15, 16, 46]. We believe that
 619 our results on TCA can help to establish complexity results for other logics (see also Section 6
 620 about a domain for strings and [33, Section 4] to handle more concrete domains).

621 ——— References ———

- 622 1 S. Abriola, D. Figueira, and S. Figueira. Logics of repeating values on data trees and branching
 623 counter systems. In *FoSSaCS'17*, volume 10203 of *LNCS*, pages 196–212, 2017.
- 624 2 E.G. Amparore, S. Donatelli, and F. Gallà. A CTL^* model checker for Petri nets. In *Petri*
 625 *Nets'20*, volume 12152 of *LNCS*, pages 403–413. Springer, 2020.
- 626 3 F. Baader and J. Rýdval. Using model theory to find decidable and tractable description
 627 logics with concrete domains. *JAR*, 66(3):357–407, 2022.
- 628 4 Ph. Balbiani and J.F. Condotta. Computational complexity of propositional linear temporal
 629 logics based on qualitative spatial or temporal reasoning. In *FroCoS'02*, volume 2309 of *LNAI*,
 630 pages 162–173. Springer, 2002.
- 631 5 K. Bartek and M. Lelyk. Modal μ -calculus with atoms. In *CSL'17*, pages 30:1–30:21.
 632 Leibniz-Zentrum für Informatik, LIPICS, 2017.
- 633 6 B. Bednarczyk and O. Fiuk. Presburger Büchi tree automata with applications to logics with
 634 expressive counting. In *WoLLIC'22*, volume 13468 of *LNCS*, pages 295–308. Springer, 2022.
- 635 7 A. Bhaskar and M. Praveen. Realizability problem for constraint LTL. In *TIME'22*, volume
 636 247 of *LIPICs*, pages 8:1–8:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.
- 637 8 M. Bojańczyk. A bounding quantifier. In *CSL'04*, volume 3210 of *LNCS*, pages 41–55. Springer,
 638 2004.
- 639 9 M. Bojańczyk and Th. Colcombet. Bounds in ω -Regularity. In *LiCS'06*, pages 285–296. IEEE
 640 Computer Society, 2006.
- 641 10 M. Bojańczyk and S. Toruńczyk. Weak MSO+U over infinite trees. In *STACS'12*, LIPICS,
 642 pages 648–660, 2012.
- 643 11 L. Bozzelli and R. Gascon. Branching-time temporal logic extended with Presburger constraints.
 644 In *LPAR'06*, volume 4246 of *LNCS*, pages 197–211. Springer, 2006.
- 645 12 L. Bozzelli and S. Pinchinat. Verification of gap-order constraint abstractions of counter
 646 systems. *TCS*, 523:1–36, 2014.
- 647 13 C. Carapelle. *On the satisfiability of temporal logics with concrete domains*. PhD thesis, Leipzig
 648 University, 2015.
- 649 14 C. Carapelle, A. Kartzow, and M. Lohrey. Satisfiability of CTL^* with constraints. In
 650 *CONCUR'13*, LNCS, pages 455–463. Springer, 2013.
- 651 15 C. Carapelle, A. Kartzow, and M. Lohrey. Satisfiability of ECTL^* with constraints. *Journal*
 652 *of Computer and System Sciences*, 82(5):826–855, 2016.
- 653 16 C. Carapelle and A.-Y. Turhan. Description Logics Reasoning w.r.t. General TBoxes is
 654 Decidable for Concrete Domains with the EHD-property. In *ECAI'16*, volume 285, pages
 655 1440–1448. IOS Press, 2016.
- 656 17 K. Čerāns. Deciding properties of integral relational automata. In *ICALP'94*, volume 820 of
 657 *LNCS*, pages 35–46. Springer, 1994.
- 658 18 A. Chandra, D. Kozen, and L. Stockmeyer. Alternation. *JACM*, 28(1):114–133, 1981.

- 659 19 R. Condurache, C. Dima, Y. Oualhdaj, and N. Troquard. Rational Synthesis in the Commons
660 with Careless and Careful Agents. In *AAMAS'21*, pages 368–376, 2021.
- 661 20 B. Cook, H. Khlaaf, and N. Piterman. On automation of CTL* verification for infinite-state
662 systems. In *CAV'15*, volume 9206 of *LNCS*, pages 13–29. Springer, 2015.
- 663 21 N. Decker, P. Habermehl, M. Leucker, and D. Thoma. Ordered navigation on multi-attributed
664 data words. In *CONCUR'14*, volume 8704 of *LNCS*, pages 497–511, 2014.
- 665 22 S. Demri and M. Deters. Temporal logics on strings with prefix relation. *Journal of Logic and
666 Computation*, 26:989–1017, 2016.
- 667 23 S. Demri and D. D'Souza. An automata-theoretic approach to constraint LTL. *I & C*,
668 205(3):380–415, 2007.
- 669 24 S. Demri and R. Gascon. Verification of qualitative \mathbb{Z} constraints. *TCS*, 409(1):24–40, 2008.
- 670 25 S. Demri and R. Lazić. LTL with the freeze quantifier and register automata. *ACM ToCL*,
671 10(3), 2009.
- 672 26 S. Demri and K. Quaas. Concrete domains in logics: a survey. *ACM SIGLOG News*, 8(3):6–29,
673 2021.
- 674 27 S. Demri and K. Quaas. Constraint automata on infinite data trees: From CTL(Z)/CTL*(Z)
675 to decision procedures. CoRR, abs/2302.05327, 2023.
- 676 28 A. Deutsch, R. Hull, and V. Vianu. Automatic verification of database-centric system. *SIGMOD
677 Record*, 43(3):5–17, 2014.
- 678 29 E.A. Emerson and J. Halpern. 'sometimes' and 'not never' revisited: on branching versus
679 linear time temporal logic. *JACM*, 33:151–178, 1986.
- 680 30 E.A. Emerson and C.S. Jutla. The complexity of tree automata and logics of programs. *SIAM
681 Journal of Computing*, 29(1):132–158, 2000.
- 682 31 E.A. Emerson and P. Sistla. Deciding full branching time logic. *Information and Control*,
683 61:175–201, 1984.
- 684 32 L. Exibard, E. Filiot, and A. Khalimov. Church synthesis on register automata over linearly
685 ordered data domains. In *STACS'21*, volume 187 of *LIPICs*, pages 28:1–28:16. Schloss Dagstuhl
686 - Leibniz-Zentrum für Informatik, 2021.
- 687 33 L. Exibard, E. Filiot, and A. Khalimov. A generic solution to register-bounded synthesis with
688 an application to discrete orders. In *ICALP'22*, volume 229 of *LIPICs*, pages 122:1–122:19.
689 Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.
- 690 34 P. Felli, M. Montali, and S. Winkler. CTL* model checking for data-aware dynamic systems
691 with arithmetic. In *IJCAR'22*, volume 13385 of *LNCS*, pages 36–56. Springer, 2022.
- 692 35 P. Felli, M. Montali, and S. Winkler. Linear-time verification of data-aware dynamic systems
693 with arithmetic. In *AAAI'22*, pages 5642–5650. AAAI Press, 2022.
- 694 36 D. Figueira. *Reasoning on words and trees with data*. PhD thesis, ENS Cachan, 2010.
- 695 37 D. Figueira. Decidability of Downward XPath. *ACM ToCL*, 13(4):1–40, 2012.
- 696 38 R. Gascon. An automata-based approach for CTL* with constraints. *Electronic Notes in
697 Theoretical Computer Science*, 239:193–211, 2009.
- 698 39 S. Göller, Ch. Haase, J. Ouaknine, and J. Worrell. Branching-time model checking of parametric
699 one-counter automata. In *FoSSaCS'12*, volume 7213 of *LNCS*, pages 406–420. Springer, 2012.
- 700 40 J.F. Groote and R. Mastescu. Verification of temporal properties of processes in a setting
701 with data. In *AMAST'98*, volume 1548 of *LNCS*, pages 74–90, 1998.
- 702 41 R. Iosif and X. Xu. Alternating automata modulo first order theories. In *CAV'19*, volume
703 11562 of *LNCS*, pages 46–63, 2019.
- 704 42 M. Jurdziński and R. Lazić. Alternating automata on data trees and XPath satisfiability.
705 *ACM ToCL*, 12(3):19:1–19:21, 2011.
- 706 43 A. Kartzow and Th. Weidner. Model checking constraint LTL over trees. *CoRR*, abs/1504.06105,
707 2015.
- 708 44 O. Kupferman, M. Vardi, and P. Wolper. An automata-theoretic approach to branching-time
709 model checking. *JACM*, 47(2):312–360, 2000.

- 710 45 N. Labai. *Automata-based reasoning for decidable logics with data values*. PhD thesis, TU
711 Wien, May 2021.
- 712 46 N. Labai, M. Ortiz, and M. Simkus. An Exptime Upper Bound for \mathcal{ALC} with integers. In
713 *KR'20*, pages 425–436. Morgan Kaufman, 2020.
- 714 47 S. Lasota and I. Walukiewicz. Alternating timed automata. *ACM ToCL*, 9(2):10:1–10:27,
715 2008.
- 716 48 A. Lechner, R. Mayr, J. Ouaknine, A. Pouly, and J. Worrell. Model checking flat freeze LTL
717 on one-counter automata. *Logical Methods in Computer Science*, 14(4), 2018.
- 718 49 C. Lutz. Interval-based temporal reasoning with general TBoxes. In *IJCAI'01*, pages 89–94.
719 Morgan-Kaufmann, 2001.
- 720 50 C. Lutz. *The Complexity of Description Logics with Concrete Domains*. PhD thesis, RWTH,
721 Aachen, 2002.
- 722 51 C. Lutz. Description logics with concrete domains—a survey. In *Advances in Modal Logics*
723 *Volume 4*, pages 265–296. King’s College Publications, 2003.
- 724 52 C. Lutz. NEXPTIME-complete description logics with concrete domains. *ACM ToCL*,
725 5(4):669–705, 2004.
- 726 53 C. Lutz and M. Milčić. A Tableau Algorithm for Description Logics with Concrete Domains
727 and General Tboxes. *JAR*, 38(1-3):227–259, 2007.
- 728 54 R. Mayr and P. Totzke. Branching-time model checking gap-order constraint systems. *Funda-*
729 *menta Informaticae*, 143(3–4):339–353, 2016.
- 730 55 D. Muller and E. Schupp. Simulating alternating tree automata by nondeterministic automata:
731 New results and new proofs of the theorems of Rabin, McNaughton and Safra. *TCS*, 141(1–
732 2):69–107, 1995.
- 733 56 F. Neven, T. Schwentick, and V. Vianu. Finite state machines for strings over infinite alphabets.
734 *ACM ToCL*, 5(3):403–435, 2004.
- 735 57 D. Peteler and K. Quaas. Deciding Emptiness for Constraint Automata on Strings with
736 the Prefix and Suffix Order. In *MFCS'22*, volume 241 of *LIPICs*, pages 76:1–76:15. Schloss
737 Dagstuhl - Leibniz-Zentrum für Informatik, 2022.
- 738 58 M.O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions*
739 *of the AMS*, 141:1–35, 1969.
- 740 59 P. Revesz. *Introduction to Constraint Databases*. Springer, New York, 2002.
- 741 60 S. Safra. *Complexity of Automata on Infinite Objects*. PhD thesis, The Weizmann Institute of
742 Science, Rehovot, 1989.
- 743 61 L. Segoufin and S. Toruńczyk. Automata based verification over linearly ordered data domains.
744 In *STACS'11*, pages 81–92, 2011.
- 745 62 H. Seidl, Th. Schwentick, and A. Muscholl. Counting in trees. In *Logic and Automata: History*
746 *and Perspectives*, volume 2 of *Texts in Logic and Games*, pages 575–612. Amsterdam University
747 Press, 2008.
- 748 63 F. Song and Z. Wu. On temporal logics with data variable quantifications: decidability and
749 complexity. *I & C*, 251:104–139, 2016.
- 750 64 W. Thomas. Automata on infinite objects. In *Handbook of Theoretical Computer Science,*
751 *Volume B, Formal models and semantics*, pages 133–191. Elsevier, 1990.
- 752 65 Sz. Toruńczyk and Th. Zeume. Register automata with extrema constraints, and an application
753 to two-variable logic. *Logical Methods in Computer Science*, 18(1), 2022.
- 754 66 M. Vardi and L. Stockmeyer. Improved upper and lower bounds for modal logics of programs.
755 In *STOC'85*, pages 240–251. ACM, 1985.
- 756 67 M. Vardi and Th. Wilke. Automata: from logics to algorithms. In *Logic and Automata:*
757 *History and Perspectives*, number 2 in *Texts in Logic and Games*, pages 629–736. Amsterdam
758 University Press, 2008.
- 759 68 M. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *Journal*
760 *of Computer and System Sciences*, 32:183–221, 1986.
- 761 69 M. Vardi and P. Wolper. Reasoning about infinite computations. *I & C*, 115:1–37, 1994.

29:18 Constraint Automata on Infinite Data Trees

- 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.