

Tree Automata with Height Constraints between Brothers

Carles Creus and Guillem Godoy ^{*}

Universitat Politècnica de Catalunya, Departament de Llenguatges i Sistemes Informàtics, Barcelona, ESP, c creuslopez@gmail.com, g godoy@lsi.upc.edu

Abstract. We define the tree automata with height constraints between brothers (TACBB_H). Constraints of equalities and inequalities between heights of siblings that restrict the applicability of the rules are allowed in TACBB_H . These constraints allow to express natural tree languages like complete or balanced (like AVL) trees. We prove decidability of emptiness and finiteness for TACBB_H , and also for a more general class that additionally allows to combine equality and disequality constraints between brothers.

Keywords: Tree-Automata, Constraints, Emptiness, Finiteness

1 Introduction

For a long time, tree automata have been regarded as a fundamental tool in many theoretical areas of computer science. These automata are a generalization of word automata that work on structured input, namely trees. Due to their good computational properties, tree automata have been used as the formalism for, e.g., describing parse trees of a context-free grammar or the well-formed terms over a sorted signature [17], to characterize the solutions of formulas in monadic second-order logic [11], and to naturally capture type formalisms for XML data [19].

Unfortunately, expressiveness of plain tree automata is limited since they only have finite memory. For instance, a set like $L = \{f(t_1, t_2) \mid t_1 \neq t_2\}$ is a typical example of language that cannot be recognized by any tree automaton. In this case, the reason is that it is not possible to check that arbitrary t_1 and t_2 are different using only a finite number of states. In order to overcome such limitation, many extensions to tree automata have been considered in recent years. One of the most studied cases consists in enhancing the automaton with the possibility to test equality and disequality of certain subterms of the input tree. This kind of automata are said to have (dis)equality constraints and, depending on how the tests are defined, they can be roughly classified as either local [18] or global [12,2]. We focus on the local case. In this model, each transition rule of the automaton has an associated Boolean combination of atomic predicates

^{*} The authors were supported by an FPU grant (first author) and the FORMALISM project (TIN2007-66523) from the Spanish Ministry of Education and Science.

of the form $p_1 = p_2$ and $p_1 \neq p_2$, for positions p_1, p_2 . Such an atomic constraint holds for a given rule application if the subterms pending at p_1 and p_2 , relative to the position where the rule is applied, are equal in the first case and different in the second one. The previous example language L can be recognized with such automata: it suffices to force the predicate $1 \neq 2$ to be tested at the root position. However, the increase in expressiveness obtained with constraints comes at the expense of other desirable properties. For instance, when arbitrary positions p_1 and p_2 are allowed in the atoms, emptiness and finiteness are undecidable for this kind of automata [18,7]. Nevertheless, several particular cases enjoy better properties and have been successfully used to tackle different problems: tree automata with constraints between brother positions allowed to prove decidability of fragments of quantifier-free formulas on one-step rewriting [4], as well as the recognizability problem for regular tree languages under particular cases of tree homomorphisms [3]; arbitrary local disequality constraints were used to prove EXPTIME-completeness of ground reducibility [5]; arbitrary local disequality constraints combined with a restricted version of local equality constraints, called HOM equalities, were used to prove EXPTIME-completeness of the recognizability problem for regular tree languages under tree homomorphisms [15,9]; and automata with arbitrary local constraints but with a bound on the maximum number of equality tests that can be performed at each branch of the input tree lead to the decidability of fragments of the first-order theory of reduction [10]. Other models originated in the study of secrecy in cryptographic protocols consider extending the automata with one tree-shaped memory, allowing the transition rules to update such memory and to perform comparisons [6,8]. These models are expressive enough to generalize certain cases of local equality constraints between brothers.

We propose a new kind of automata with local constraints between brother positions that differs from the previous literature in that, instead of checking subterms for either syntactic equality or some notion of equivalence (like, e.g., in [16,2]), the restrictions are imposed on the *height* of the subterms involved in the constraints. We call them tree automata with height constraints between brothers, TACBB_H for short. More precisely, our basic atomic predicates are of the form $h(i) = h(j)$, for positions i and j of length 1, and are satisfied when the subterms pending at i and j , relative to the application of the rule, have identical height. We also consider inequality predicates of the form $h(i) < h(j)$ and comparisons introducing an integer constant x of the form $h(i) = h(j) + x$ and $h(i) < h(j) + x$, with the straightforward interpretations. It is easy to see that our notion of constraints is incomparable with syntactic equality constraints. For instance, the language of complete trees over a signature with two constant symbols a, b and a binary symbol f can be recognized by a TACBB_H with the transition rules

$$a \rightarrow q, b \rightarrow q, f(q, q) \xrightarrow{h(1)=h(2)} q$$

but such language cannot be recognized by the automata with (dis)equality constraints between brothers (AWCBB) from [3]. Intuitively, this is because, even if two terms t_1 and t_2 were inductively guaranteed to be complete, it is not pos-

sible to check with only (dis)equality constraints whether t_1 and t_2 have equal height, and thus, whether $f(t_1, t_2)$ is also complete. Actually, it can be proved that with **AWCBB** it is only possible to recognize the language of complete trees when the underlying signature has a single constant symbol and a single non-constant symbol. This limitation also holds for generalizations of **AWCBB** like the **TAB** from [2], where equalities are interpreted modulo flat theories. Another interesting language that can be recognized by **TACBB_H** using the inequality predicates on height is the language of AVL trees, i.e., the set of trees where the heights of the two direct children of any internal node differ by at most one. It suffices to replace the previous rule for f with

$$f(q, q) \xrightarrow{(h(1) < h(2) + 2) \wedge (h(2) < h(1) + 2)} q$$

Note that the constraint forces the height of each child to be at most one more than the height of the other child.

We prove decidability of the emptiness and finiteness problems for **TACBB_H**. Our approach consists in transforming the automaton into a normalized form, and obtaining a recursive formulation to describe the set of reachable states when recognizing terms of a specific height. The decision algorithm follows directly from such result. We adapt the method to automata that combine the height constraints of **TACBB_H** with the (dis)equality constraints of **AWCBB**, which we call **TACBB_{He}**. Note that this kind of automata strictly generalizes **TACBB_H** and **AWCBB**, and hence, decidability of emptiness and finiteness is EXPTIME-hard [7].

The definition of **TACT** in [20] is incomparable with **TACBB_{He}**. Nevertheless, a given tree automaton with height constraints is transformable into a **TACT** by preserving emptiness (but not the language). But this does not help in our setting to decide emptiness of our model, since emptiness of **TACT** is undecidable. The definition of **VTAM_R^R** in [8] captures our automata models. Nevertheless, emptiness is only decidable for some particular subclasses that are incomparable with **TACBB_{He}** since, although they can recognize the particular set of complete trees, height of subtrees cannot be compared in general.

The paper is structured as follows. In Section 2 we introduce standard notations and definitions, and in Section 3 we define our model of automata. In Section 4 we tackle the case where only simple height constraints are allowed, i.e., **TACBB_H** whose atomic predicates are restricted to $h(i) = h(j)$ and $h(i) < h(j)$. In Section 5 we generalize those results to **TACBB_{He}**. We conclude in Section 6.

2 Preliminaries

The size of a finite set S is denoted by $|S|$, and the powerset of S is denoted by 2^S . Given two sets of sets S_1, S_2 , by $S_1 \boxtimes S_2$ we denote the set of sets that are obtained by choosing a set $\hat{S}_1$ from S_1 and a set $\hat{S}_2$ from S_2 and making their union, i.e., $\{\hat{S}_1 \cup \hat{S}_2 \mid \hat{S}_1 \in S_1 \wedge \hat{S}_2 \in S_2\}$. For example, $\{\{a, b\}, \{c\}\} \boxtimes \{\{d, e\}, \{f\}\} = \{\{a, b, d, e\}, \{a, b, f\}, \{c, d, e\}, \{c, f\}\}$. A partition P of a set S is a set of non-empty sets (the parts) that are pairwise disjoint and whose union is S . By $i \equiv_P j$ we denote that i, j belong to the same part of P .

We introduce notation for terms and positions, for a survey see [1]. A *signature* consists of an alphabet Σ , i.e., a finite set of symbols, together with a mapping that assigns to each symbol in Σ a natural number (possibly 0), its *arity*. We denote by $\Sigma^{(m)}$ the subset of symbols in Σ of arity m , and by $\maxar(\Sigma)$ the maximum m such that $\Sigma^{(m)}$ is not empty. We write simply $\maxar$ when Σ is known by the context. Symbols in $\Sigma^{(0)}$, called *constants*, are denoted by a, b , with possible subscripts. The set $\mathcal{T}(\Sigma)$ of *ground terms* (or just terms) over Σ is the smallest set such that $f(t_1, \dots, t_m)$ is in $\mathcal{T}(\Sigma)$ whenever $f \in \Sigma^{(m)}$, and $t_1, \dots, t_m \in \mathcal{T}(\Sigma)$. For a term of the form $a()$ we simply write a . A *language* over Σ is a set of ground terms. A *position* is a sequence of natural numbers. The symbol λ denotes the empty sequence, also called the root position, and $p.p'$ denotes the concatenation of the positions p and p' . The set of positions of a term t , denoted $\text{Pos}(t)$, is defined recursively as $\text{Pos}(f(t_1, \dots, t_m)) = \{\lambda\} \cup \{i.p \mid i \in \{1, \dots, m\} \wedge p \in \text{Pos}(t_i)\}$. The *subterm* of a term t at a position p , denoted $t|_p$, is defined recursively as $t|_\lambda = t$ and $f(t_1, \dots, t_m)|_{i.p} = t_i|_p$. The *height* of a term t , denoted $\text{height}(t)$, is defined recursively as $\text{height}(t) = 0$ if t is a constant, and as $\text{height}(f(t_1, \dots, t_m)) = 1 + \max\{\text{height}(t_1), \dots, \text{height}(t_m)\}$ otherwise.

Tree automata and regular tree languages are well-known concepts of theoretical computer science [13,14,7]. Here we only recall the notion of tree automata.

Definition 1. A tree automaton, TA for short, is a tuple $A = \langle Q, \Sigma, F, \Delta \rangle$, where Q is a finite set of states, Σ is a signature, $F \subseteq Q$ is the set of final states, and Δ is a set of rules of the form $f(q_1, \dots, q_m) \rightarrow q$, where $q_1, \dots, q_m, q$ are in Q , and f is in $\Sigma^{(m)}$.

A run of A on a term $t \in \mathcal{T}(\Sigma)$ is a mapping $r : \text{Pos}(t) \rightarrow \Delta$ such that, for each position $p \in \text{Pos}(t)$, if $t|_p$ is of the form $f(t_1, \dots, t_m)$, then $r(p)$ is a rule of the form $f(q_1, \dots, q_m) \rightarrow q$, and the rules $r(p.1), \dots, r(p.m)$ have $q_1, \dots, q_m$ as right-hand sides, respectively. We say that $r(p)$ is the rule applied at position p . The run r is called accepting if the right-hand side of $r(\lambda)$ is a state in F . A term t is accepted by A if there exists an accepting run of A on t . The set of accepted terms by A , also called the language recognized by A , is denoted by $\mathcal{L}(A)$. A language L is regular if there exists a TA A such that $\mathcal{L}(A) = L$ holds.

3 Tree automata with height and equality constraints between brothers

In this paper we deal with several classes of tree automata with constraints between brothers. The only difference between such classes is the form of the constraints and the way such constraints are interpreted. In this section we give a general definition of tree automata with constraints between brothers that is instantiated later. To simplify notations, we consider a fixed signature Σ .

Definition 2. A constraint structure is a tuple $\langle C, \text{PosVar}, \models, |\cdot|, \text{Width} \rangle$. Here, C (notion of syntax) is a set of elements called constraints. PosVar is a function that maps each element of C to a finite set of natural numbers. Given $c \in C$, any partial mapping $I : \mathbb{N} \rightarrow \mathcal{T}(\Sigma)$ satisfying $\text{PosVar}(c) \subseteq \text{Dom}(I)$ is called

an interpretation of c . $\models$ (notion of satisfaction) maps each pair $c \in C$ and interpretation I of c to either true (denoted $I \models c$) or false (denoted $I \not\models c$). $|\cdot|$ (notion of size) maps each constraint c to a natural number. Width maps each constraint c to a natural number.

Given two constraints c_1, c_2 (not necessarily from the same constraint structure) holding $\text{PosVar}(c_1) = \text{PosVar}(c_2)$, we say that c_1, c_2 are compatible if there exists I satisfying $I \models c_1$ and $I \models c_2$ (where $\models$ refers to the satisfaction notion of each corresponding structure). Otherwise, we say that c_1, c_2 are incompatible.

A set of constraints $\{c_1, \dots, c_n\}$ of the same constraint structure and holding $\text{PosVar}(c_1) = \dots = \text{PosVar}(c_n)$ is called deterministic if all c_i, c_j are pairwise incompatible for $1 \leq i < j \leq n$. It is called complete if, for each $I : \text{PosVar}(c_1) \rightarrow \mathcal{T}(\Sigma)$, $I \models c_i$ for some $1 \leq i \leq n$.

Given two constraints c_1, c_2 (not necessarily from the same constraint structure) holding $\text{PosVar}(c_1) \supseteq \text{PosVar}(c_2)$, we say that c_1 implies c_2 , denoted $c_1 \models c_2$, if for each I satisfying $I \models c_1$, $I \models c_2$ also holds (where $\models$ refers to the satisfaction notion of each corresponding structure).

Definition 3. Let $S = \langle C, \text{PosVar}, \models, |\cdot|, \text{Width} \rangle$ be a constraint structure. A tree automaton with constraints between brothers based on S , TACBB_S for short, is a tuple $A = \langle Q, \Sigma, F, \Delta \rangle$, where Q is a finite set of states, Σ is a signature, $F \subseteq Q$ is the set of final states, and Δ is a set of rules of the form $f(q_1, \dots, q_m) \xrightarrow{c} q$, where $q_1, \dots, q_m, q$ are in Q , f is in $\Sigma^{(m)}$, and c is a constraint of C satisfying $\text{PosVar}(c) \subseteq \{1, \dots, m\}$. The rule is fully constrained if $\text{PosVar}(c) = \{1, \dots, m\}$. The size of such rule is $m + 2 + |c|$. The size of A is $|Q|$ plus the sum of sizes of all rules in Δ . The width of a rule $f(q_1, \dots, q_m) \xrightarrow{c} q$ of A is $\text{Width}(c)$. The width of A is the maximum among the widths of its rules. A is fully constrained, denoted TACBB_S^F , if each of its rules is fully constrained.

A run of A on a term $t \in \mathcal{T}(\Sigma)$ is a mapping $r : \text{Pos}(t) \rightarrow \Delta$ such that, for each position $p \in \text{Pos}(t)$, if $t|_p$ is of the form $f(t_1, \dots, t_m)$, then $r(p)$ is a rule of the form $f(q_1, \dots, q_m) \xrightarrow{c} q$, the rules $r(p.1), \dots, r(p.m)$ have $q_1, \dots, q_m$ as right-hand sides, respectively, and $I = \{1 \mapsto t|_{p.1}, \dots, m \mapsto t|_{p.m}\} \models c$. We say that $r(p)$ is the rule applied at position p . The run r is called accepting if the right-hand side of $r(\lambda)$ is a state in F . A term t is accepted by A if there exists an accepting run of A on t . The set of accepted terms by A , also called the language recognized by A , is denoted by $\mathcal{L}(A)$. By $\mathcal{L}(A, q)$ we denote the set of terms for which there exists a run r of A on them holding that the right-hand side of $r(\lambda)$ is q .

Definition 4. A height and equality constraint c is a boolean combination (including negation) of atoms of the form either $h(i) = h(j)$ or $h(i) < h(j)$ or $h(i) = h(j) + x$ or $h(i) < h(j) + x$ or $i = j$ for natural numbers i, j and integer number x . The width of c , denoted $\text{Width}(c)$, is 1 plus the maximum of the absolute values of such x . The size of c , denoted $|c|$, is the number of such atoms occurring in c . By $\text{PosVar}(c)$ we denote the set of such naturals i, j occurring in c . An interpretation of c is a partial mapping $I : \mathbb{N} \rightarrow \mathcal{T}(\Sigma)$ such that $\text{PosVar}(c) \subseteq \text{Dom}(I)$. I satisfies (is a solution of) c , denoted $I \models c$ if, by

replacing in c each i by $I(i)$, the expression evaluates to true by interpreting h as the height function, $=$ as syntactic equality when it compares terms or as equality of the evaluated expressions when it compares integer numbers, $<$, $+$ as the less and addition operators on integer numbers, and the boolean operators in the usual way.

By C_{He} we denote the set of all height and equality constraints. By C_H we denote the subset of C_{He} of constraints with no occurrences of atoms of the form $i = j$ (i.e., only height constraints). By C_{he} we denote the subset of C_{He} whose atoms are of the form $h(i) = h(j)$ or $h(i) < h(j)$ or $i = j$ (i.e., equality constraints and only simple height constraints). By C_h we denote the subset of C_{he} whose atoms are of the form $h(i) = h(j)$ or $h(i) < h(j)$ (i.e., only simple height constraints). The constraint structures He , he , H , h are defined as $\langle C, \text{PosVar}, \models, |\cdot|, \text{Width} \rangle$, by replacing C by C_{He} , C_{he} , C_H , C_h , respectively, and where $\text{PosVar}, \models, |\cdot|, \text{Width}$ are defined as above.

We will show that emptiness and finiteness of TACBB_{He} are decidable, obtaining different time complexities depending on the restrictions of the constraints. For explanation purposes, we deal with the most particular case TACBB_h in Section 4, and the most general case TACBB_{He} in Section 5, where the results for TACBB_H and TACBB_{he} are also stated.

4 The case with simple height constraints

In this section we prove decidability of emptiness and finiteness of TACBB_h . A usual way to deal with automata with constraints is to transform them into a normalized form that is easy to deal with. We proceed in this way by normalizing simple height constraints according to the following definitions and lemma.

Definition 5. A normalized simple height constraint is an expression c of the form $S_1 < S_2 < \dots < S_n$, where all $S_1, \dots, S_n$ are non-empty finite sets of natural numbers, and each natural number occurs in at most one S_i . By $\text{PosVar}(c)$ we denote the set of natural numbers occurring in $S_1 \uplus \dots \uplus S_n$. An interpretation of c is a partial mapping I from natural numbers to terms in $\mathcal{T}(\Sigma)$, whose domain includes $\text{PosVar}(c)$. We say that I satisfies (is a solution of) c , denoted $I \models c$ if:

- for each $i \in \{1, \dots, n\}$, the natural numbers occurring in S_i are mapped by I to terms with the same height, and
- for each $i \in \{1, \dots, n-1\}$, each $i_1 \in S_i$ and each $i_2 \in S_{i+1}$, $\text{height}(I(i_1)) < \text{height}(I(i_2))$.

The size of c is $|c| = |\text{PosVar}(c)|$. The width of c is $\text{Width}(c) = 1$ (in the simple case the width plays no important role). The structure of normalized simple height constraints is the constraint structure $nh = \langle C_{nh}, \text{PosVar}, \models, |\cdot|, \text{Width} \rangle$, where C_{nh} is the set of all normalized simple height constraints, and $\text{PosVar}, \models, |\cdot|, \text{Width}$ are defined as above.

Lemma 1. Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a TACBB_h^F . Then, a $\text{TACBB}_{nh}^F A' = \langle Q, \Sigma, F, \Delta' \rangle$ satisfying $\mathcal{L}(A') = \mathcal{L}(A)$ can be computed in time $\mathcal{O}(|A| \cdot 2^{\maxar^2})$.

Proof. Sketch: it suffices to define Δ' as $\{f(q_1, \dots, q_m) \xrightarrow{c'} q \mid c' \in C_{nh} \wedge \text{PosVar}(c') = \{1, \dots, m\} \wedge \exists (f(q_1, \dots, q_m) \xrightarrow{c} q) \in \Delta : (c' \models c)\}$.

To decide emptiness of a $\text{TACBB}_{nh}^F A$, we will need to compute, for each h , which states are reachable by runs of A on terms with height h . To this end, we need to determine which rules can be applied at the root of such terms, and this leads to a more general computation: we need to know which part of a constraint is satisfied by taking into account runs on terms smaller than h .

Definition 6. Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a TACBB_{nh}^F . Let h be a natural number. Let q be a state in Q . Then, we define $\text{ExistTerm}_A(h, q)$ as true if there exists a term with height h in $\mathcal{L}(A, q)$, and as false otherwise. Let c be a normalized simple height constraint. Let I be a solution of c . Let N be a partial mapping from natural numbers to Q such that $\text{Dom}(N) \supseteq \text{Dom}(I)$. We say that N and I are compatible with respect to A if, for each $i \in \text{Dom}(I)$, $I(i) \in \mathcal{L}(A, N(i))$. We define $\text{ExistSol}_A(h, c, N)$ as true if there exists $I : \text{PosVar}(c) \rightarrow \mathcal{T}(\Sigma)$ such that I is compatible with N and a solution of c , and the highest term occurring in $I(\text{PosVar}(c))$ has height h , and as false otherwise. Moreover, we define $\text{AccExistSol}_A(h, c, N)$ as $\bigvee_{h' \leq h} \text{ExistSol}_A(h', c, N)$. When A is clear from the context, we omit the subindex of ExistTerm_A , ExistSol_A , AccExistSol_A .

We define C_A as the set of pairs $\{\langle c, N \rangle \mid \exists (f(q_1, \dots, q_m) \xrightarrow{c'} q) \in \Delta : (c \text{ is a non-empty prefix of } c' \text{ or a set occurring in } c') \wedge N = \{1 \mapsto q_1, \dots, m \mapsto q_m\}\}$. Note that $|C_A|$ is bounded by $2 \cdot \maxar \cdot |\Delta|$. The configuration of A for height h is the description of all values $\text{ExistSol}_A(h, c, N)$, $\text{AccExistSol}_A(h, c, N)$, $\text{ExistTerm}_A(h, q)$ for $\langle c, N \rangle \in C_A$ and $q \in Q$.

The following lemma gives a computable definition of ExistSol , AccExistSol and ExistTerm , and a way to argue that the configuration for $h > 0$ depends only on the configuration for $h - 1$. It can be proved by induction on h and $|c|$.

Lemma 2. Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a TACBB_{nh}^F . ExistSol , AccExistSol and ExistTerm can also be defined recursively as follows:

- Consider the case $h = 0$. Then, $\text{ExistTerm}(h, q)$ is true if and only if there exists $a \in \Sigma^{(0)}$ satisfying $a \in \mathcal{L}(A, q)$. Moreover, if c is not just a set, then $\text{ExistSol}(h, c, N) = \text{AccExistSol}(h, c, N) = \text{False}$, and otherwise if c is a set S , $\text{ExistSol}(h, c, N)$ and $\text{AccExistSol}(h, c, N)$ are defined to be true if and only if $\bigwedge_{q \in N(S)} \text{ExistTerm}(h, q)$ holds. In the rest of cases assume $h > 0$.
- $\text{AccExistSol}(h, c, N) = \text{AccExistSol}(h - 1, c, N) \vee \text{ExistSol}(h, c, N)$
- $\text{ExistSol}(h, c' < S, N) = \text{AccExistSol}(h - 1, c', N) \wedge \text{ExistSol}(h, S, N)$
- $\text{ExistSol}(h, S, N) = \bigwedge_{q \in N(S)} \text{ExistTerm}(h, q)$
- $\text{ExistTerm}(h, q) = \bigvee_{(f(q_1, \dots, q_m) \xrightarrow{c} q) \in \Delta, m > 0} \text{ExistSol}(h - 1, c, \{1 \mapsto q_1, \dots, m \mapsto q_m\})$

Corollary 1. *Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a $\text{TACBB}_{\text{nh}}^F$. Let h be a natural number. The configuration of A for height $h+1$ can be obtained from the configuration of A for height h with time in $\mathcal{O}(|C_A| \cdot \text{maxar} + |Q| + |\Delta|)$, i.e., in $\mathcal{O}(\text{maxar}^2 \cdot |\Delta| + |Q|)$.*

The previous result leads to the fact that, when we find the same configuration for heights $h_1 < h_2$, the sequence of configurations is periodic with period $h_2 - h_1$ starting from h_1 . Thus, deciding emptiness corresponds to check whether there exists an accepting state q satisfying $\text{ExistTerm}(h, q)$ for $h < h_2$, and deciding finiteness corresponds to check whether there exists an accepting state q satisfying $\text{ExistTerm}(h, q)$ for $h_1 \leq h < h_2$. The final time complexity depends on Corollary 1 and the number of different possible configurations.

Lemma 3. *Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a $\text{TACBB}_{\text{nh}}^F$. The number of different configurations of A considering all possible h is bounded by $2^{|Q|} \cdot (2 \cdot \text{maxar} \cdot |\Delta| + 1)$.*

Proof. Consider any two heights $0 < h_1 < h_2$ satisfying that AccExistSol for $h_1 - 1$ is equal to AccExistSol for $h_2 - 1$, and ExistTerm for h_1 is equal to ExistTerm for h_2 . Since ExistSol for any height $h > 0$ depends only on AccExistSol for $h - 1$ and ExistTerm for h , it follows that ExistSol for h_1 is equal to ExistSol for h_2 . Moreover, it also follows that AccExistSol for h_1 is equal to AccExistSol for h_2 . Thus, the configurations of A for h_1 and h_2 are equal, and hence, to prove the statement it suffices to bound the number of different combinations of AccExistSol and ExistTerm . To this end, note that there are at most $2^{|Q|}$ different ExistTerm , and since $\text{AccExistSol}(h, c, N) \Rightarrow \text{AccExistSol}(h + 1, c, N)$ holds for any h and $\langle c, N \rangle \in C_A$, there are at most $|C_A| + 1 = 2 \cdot \text{maxar} \cdot |\Delta| + 1$ different AccExistSol .

Corollary 2. *Emptiness and finiteness of the language recognized by a $\text{TACBB}_{\text{nh}}^F$ $A = \langle Q, \Sigma, F, \Delta \rangle$ are decidable with time in $2^{\mathcal{O}(|Q| + \log(\text{maxar} \cdot |\Delta|))}$.*

As a consequence of Lemma 1 we also obtain decidability for TACBB_h .

Corollary 3. *Emptiness and finiteness of the language recognized by a TACBB_h $A = \langle Q, \Sigma, F, \Delta \rangle$ are decidable with time in $2^{\mathcal{O}(|Q| + \text{maxar}^2 + \log(|A|))}$.*

5 Decidability of the general case

The global approach for proving decidability of emptiness and finiteness for the general case of height and equality constraints is analogous to the case of simple height constraints. We start with a process of normalization of the automata.

Definition 7. *A normalized height and equality constraint is an expression c of the form $P_1 \otimes_1 P_2 \otimes_2 \cdots \otimes_{n-1} P_n$, where $P_1, \dots, P_n$ are partitions of non-empty finite sets of natural numbers, each natural number occurs in at most one P_i , and each $\otimes_i$ is an operator of the form either $=_h$ or $\leq_h$ for a natural number $h > 0$. By $\text{PosVar}(c)$ we denote the set of natural numbers occurring in $P_1, \dots, P_n$. An interpretation of c is a partial mapping $I : \mathbb{N} \rightarrow \mathcal{T}(\Sigma)$ whose domain includes $\text{PosVar}(c)$. We say that I satisfies (is a solution of) c , denoted $I \models c$, if:*

- for each $i \in \{1, \dots, n\}$, the natural numbers occurring in P_i are mapped to terms with the same height, and two natural numbers are mapped to the same term if and only if they occur in the same part, and
- for each $i \in \{1, \dots, n-1\}$ such that $\otimes_i$ is of the form $=_h$, each $i_1 \in \text{PosVar}(P_i)$ and each $i_2 \in \text{PosVar}(P_{i+1})$, $\text{height}(I(i_1)) + h = \text{height}(I(i_2))$, and
- for each $i \in \{1, \dots, n-1\}$ such that $\otimes_i$ is of the form $\leq_h$, each $i_1 \in \text{PosVar}(P_i)$ and each $i_2 \in \text{PosVar}(P_{i+1})$, $\text{height}(I(i_1)) + h \leq \text{height}(I(i_2))$.

The size of c is $|c| = |\text{PosVar}(c)|$. The width of c , denoted $\text{Width}(c)$, is the maximum among the natural numbers h occurring in the subscripts of the operators in c . The structure of normalized height and equality constraints is the constraint structure $\text{nHe} = \langle C_{\text{nHe}}, \text{PosVar}, \models, |\cdot|, \text{Width} \rangle$, where C_{nHe} is the set of all normalized height and equality constraints, and $\text{PosVar}, \models, |\cdot|, \text{Width}$ are defined as above.

Since we will need to compute not only if a state q is reachable by some term with height h , but also how many of such different terms reach q , we will determinize the automaton in order to ease this task.

Definition 8. We say that a $\text{TACBB}_{\text{nHe}}^F A = \langle Q, \Sigma, F, \Delta \rangle$ is deterministic and complete (or a $\text{dcTACBB}_{\text{nHe}}^F$) if:

- for each $f \in \Sigma$ with arity m and states $q_1, \dots, q_m \in Q$, and each normalized height and equality constraint c , there is at most one $q \in Q$ holding $(f(q_1, \dots, q_m) \xrightarrow{c} q) \in \Delta$, and
- for each $f \in \Sigma$ with arity m and states $q_1, \dots, q_m \in Q$, the set $\{c \mid \exists q \in Q : (f(q_1, \dots, q_m) \xrightarrow{c} q) \in \Delta\}$ is non-empty, deterministic, and complete.

Note that, for each $\text{dcTACBB}_{\text{nHe}}^F A$ and each $t \in \mathcal{T}(\Sigma)$, there exists one and only one run of A on t . We denote by $A(t)$ the state reached by such run at λ . In order to obtain a deterministic and complete set of constraints, we will construct them from a limited number of operators.

Definition 9. Let m, w be natural numbers. A normalized constraint with respect to m and w is a normalized height and equality constraint c over the operators $=_1, =_2, \dots, =_{w-1}$ and $\leq_w$ and satisfying $\text{PosVar}(c) = \{1, \dots, m\}$.

For a set of constraints to be deterministic, it must satisfy that its constraints are pairwise incompatible.

Lemma 4. Let m, w be natural numbers. Let c, d be two different normalized constraints with respect to m and w . Then, c and d are incompatible.

For a set of constraints to be complete, it must satisfy that any interpretation is a solution of at least one of the constraints of the set.

Definition 10. Let w be a natural number. Let $I : \mathbb{N} \rightarrow \mathcal{T}(\Sigma)$ be a partial function holding $\text{Dom}(I) = \{1, \dots, m\}$ for some m . Let $h_1 < h_2 < \dots < h_n$ be the elements of $\text{height}(I(\{1, \dots, m\}))$. The constraint induced from I and w is the normalized constraint $P_1 \otimes_1 P_2 \otimes_2 \dots \otimes_{n-1} P_n$ with respect to m and w , where:

- for $i \in \{1, \dots, n\}$, P_i is a partition of $\{j \in \{1, \dots, m\} \mid \text{height}(I(j)) = h_i\}$ satisfying $(j_1 \equiv_{P_i} j_2) \Leftrightarrow (I(j_1) = I(j_2))$, and
- for $i \in \{1, \dots, n-1\}$, $\otimes_i$ is $=_{h_{i+1}-h_i}$ if $h_{i+1} - h_i < w$, and $\leq_w$ otherwise.

Given w and an interpretation I , the fact that the constraint c induced from I and w holds $I \models c$ follows by definition.

Lemma 5. Let w be a natural number. Let $I : \mathbb{N} \rightarrow \mathcal{T}(\Sigma)$ be a partial function holding $\text{Dom}(I) = \{1, \dots, m\}$ for some m . Let c be the constraint induced from I and w . Then, $I \models c$.

Corollary 4. Let m, w be natural numbers. Let C be the set of normalized constraints with respect to m and w . Then, C is deterministic and complete.

Proof. By Lemma 4, any two different constraints in C are incompatible, and thus C is deterministic. By Lemma 5, any interpretation $I : \{1, \dots, m\} \rightarrow \mathcal{T}(\Sigma)$ satisfies the constraint c induced from I and w . Since such c is in C by definition, it follows that C is complete.

In order to normalize a $\text{TACBB}_{\text{He}} A$, we construct a $\text{dcTACBB}_{\text{nHe}}^F A'$ that simulates all possible runs of A . To this end, each rule of A' needs to determine exactly which rules of A are applicable. In particular, A' needs to determine which constraints are satisfied. For this reason, we will need to use the fact that the normalized constraints are, in some sense, more precise than the original ones, that is, given a normalized constraint c_1 and an original constraint c_2 , either c_1 and c_2 are incompatible, or $c_1 \models c_2$ holds.

Lemma 6. Let m, w be natural numbers. Let c_1 be a normalized constraint with respect to m and w . Let c_2 be a height and equality constraint whose width is smaller than or equal to w and satisfying $\text{PosVar}(c_2) \subseteq \{1, \dots, m\}$. Then, either c_1 and c_2 are incompatible, or $c_1 \models c_2$ holds.

Proof. Sketch: it suffices to note that for any two solutions I_1, I_2 of c_1 and any atom c'_2 occurring in c_2 , either $I_1 \models c'_2$ and $I_2 \models c'_2$, or $I_1 \not\models c'_2$ and $I_2 \not\models c'_2$.

Definition 11. Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a TACBB_{He} . Let w be the width of A . For each natural number m , let C_m be the set of normalized constraints with respect to m and w . Then, the normalization of A is defined as the $\text{TACBB}_{\text{nHe}}^F \langle 2^Q, \Sigma, \{F' \in 2^Q \mid F' \cap F \neq \emptyset\}, \Delta' \rangle$, where Δ' is the set of rules $\{f(S_1, \dots, S_m) \xrightarrow{c} S \mid f \in \Sigma^{(m)} \wedge c \in C_m \wedge S_1, \dots, S_m, S \in 2^Q \wedge S = \{q \in Q \mid \exists q_1 \in S_1, \dots, q_m \in S_m, c' : ((f(q_1, \dots, q_m) \xrightarrow{c'} q) \in \Delta \wedge (c \models c')))\}$.

Lemma 7. *Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a TACBB_{He} having width w . Then, the normalization A' of A is deterministic and complete, i.e., a $\text{dcTACBB}_{\text{nHe}}^F$, and can be computed with time in $2^{\mathcal{O}(\log(\Sigma) + \text{maxar} \cdot (|Q| + \text{maxar} + \log(w)) + \log(|A|))}$.*

Proof. The fact that A' is a $\text{dcTACBB}_{\text{nHe}}^F$ follows from Definition 8, Corollary 4 and Definition 11. The time complexity follows from these observations: each rule of A' can be computed with time in $\mathcal{O}(|A|)$, the number of rules of A' is bounded by $|\Sigma| \cdot 2^{|Q| \cdot \text{maxar}} \cdot |C_{\text{maxar}}|$, where C_{maxar} is the set of normalized constraints with respect to maxar and w , and $|C_{\text{maxar}}|$ is in $2^{\mathcal{O}(\text{maxar}^2 + \text{maxar} \cdot \log(w))}$.

In order to see that A' preserves the language recognized by A , we show that, for any term t , the unique state $A'(t)$ is precisely the set of states that are reachable by runs of A on t .

Lemma 8. *Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a TACBB_{He} . Let A' be the normalization of A . Let $t \in \mathcal{T}(\Sigma)$ be a term. Then, the state reached at λ by the unique run of A' on t is $S = \{q \in Q \mid t \in \mathcal{L}(A, q)\}$.*

Proof. We prove it by induction on $\text{height}(t)$. Without loss of generality, let t be of the form $f(t_1, \dots, t_m)$. By induction hypothesis, the states reached at λ by the unique runs of A' on $t_1, \dots, t_m$ are $S_1, \dots, S_m$, respectively, where $S_i = \{q \in Q \mid t_i \in \mathcal{L}(A, q)\}$. Let c be the constraint induced from $I = \{1 \mapsto t_1, \dots, m \mapsto t_m\}$ and the width of A . By Definition 11, A' has the rule $f(S_1, \dots, S_m) \xrightarrow{c} S'$ where $S' = \{q \in Q \mid \exists q_1 \in S_1, \dots, q_m \in S_m, c' : ((f(q_1, \dots, q_m) \xrightarrow{c'} q) \in \Delta \wedge (c \models c'))\}$. Such rule can be applied at the root of t since $I \models c$ holds by Lemma 5, and moreover, by Corollary 4 no other rule can. It remains to prove $S' = S$:

- ($\subseteq$) Let $q \in S'$. By definition there are $q_1 \in S_1, \dots, q_m \in S_m$ and c' such that $(f(q_1, \dots, q_m) \xrightarrow{c'} q) \in \Delta$ and $c \models c'$. Thus, $I \models c'$, and since we had that $t_1 \in \mathcal{L}(A, q_1), \dots, t_m \in \mathcal{L}(A, q_m)$, it follows $t \in \mathcal{L}(A, q)$, and hence, $q \in S$.
- ($\supseteq$) Let $q \in S$. By definition there is a run of A on t with a rule of the form $f(q_1, \dots, q_m) \xrightarrow{c'} q$ applied at the root. Note that $I \models c'$, and since $I \models c$, by Lemma 6, $c \models c'$. Since we had that $q_1 \in S_1, \dots, q_m \in S_m$, it follows $q \in S'$.

Corollary 5. *Let A be a TACBB_{He} . Let A' be the normalization of A . Then, $\mathcal{L}(A') = \mathcal{L}(A)$.*

To decide emptiness of a $\text{dcTACBB}_{\text{nHe}}^F$ A , we will need to compute, for certain h 's, how many runs (up to a certain bound) of A on terms with height h reach each state. We start by a previous definition describing which terms (and not only how many of them) are reached, in order to ease later arguments.

Definition 12. *Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a $\text{dcTACBB}_{\text{nHe}}^F$. Let h be a natural number. Let q be a state in Q . Then, $\text{Terms}_A(h, q)$ is defined as $\{t \in \mathcal{T}(\Sigma) \mid A(t) = q \wedge \text{height}(t) = h\}$. Let c be a normalized height and equality constraint. Let I be a solution of c . Let N be a partial mapping from natural numbers to Q such that $\text{Dom}(N) \supseteq \text{Dom}(I)$. We say that N and I are compatible with respect to*

A if, for each $i \in \text{Dom}(I)$, $A(I(i)) = N(i)$. We define $\text{Sols}_A(h, c, N)$ as the set of interpretations $I : \text{PosVar}(c) \rightarrow \mathcal{T}(\Sigma)$ such that I is compatible with N and a solution of c , and the highest term occurring in $I(\text{PosVar}(c))$ has height h . Moreover, we define $\text{AccSols}_A(h, c, N)$ as $\bigcup_{h' \leq h} \text{Sols}_A(h', c, N)$. When A is clear from the context, we omit the subindex of Terms_A , Sols_A , AccSols_A .

We define C_A as the set of pairs $\{\langle c, N \rangle \mid \exists (f(q_1, \dots, q_m) \xrightarrow{c'} q) \in \Delta : (c \text{ is a non-empty prefix of } c' \text{ or a partition occurring in } c') \wedge N = \{1 \mapsto q_1, \dots, m \mapsto q_m\}\}$. Note that $|C_A|$ is bounded by $2 \cdot \text{maxar} \cdot |\Delta|$.

The following lemma provides a computable definition of the sets Sols , AccSols and Terms , as well as a way to argue that the configuration for a certain h depends only on the configurations for $h - i$, with i bounded by the width of A . It can be straightforwardly proved by induction on h and on the size of c .

Lemma 9. *Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a $\text{dcTACBB}_{\text{nHe}}^F$. Sols , AccSols and Terms can also be defined recursively as follows:*

- If $h < 0$, then $\text{Terms}(h, q) = \text{Sols}(h, c, N) = \text{AccSols}(h, c, N) = \emptyset$ for any q, c, N .
- Consider the case $h = 0$. Then, $\text{Terms}(h, q) = \{a \in \Sigma^{(0)} \mid A(a) = q\}$. Moreover, if c is not just a partition, then $\text{Sols}(h, c, N) = \text{AccSols}(h, c, N) = \emptyset$, and otherwise if c is a partition P , $\text{Sols}(h, c, N)$ and $\text{AccSols}(h, c, N)$ are $\{I : \text{PosVar}(P) \rightarrow \Sigma^{(0)} \mid \forall i \in \text{PosVar}(P) : I(i) \in \text{Terms}(h, N(i)) \wedge \forall i, j \in \text{PosVar}(P) : (I(i) = I(j) \Leftrightarrow i \equiv_P j)\}$. In the rest of cases assume $h > 0$.
- $\text{AccSols}(h, c, N) = \text{AccSols}(h - 1, c, N) \cup \text{Sols}(h, c, N)$
- $\text{Sols}(h, c' =_{h'} P, N) = \text{Sols}(h - h', c', N) \boxtimes \text{Sols}(h, P, N)$
- $\text{Sols}(h, c' \leq_{h'} P, N) = \text{AccSols}(h - h', c', N) \boxtimes \text{Sols}(h, P, N)$
- $\text{Sols}(h, P, N) = \{I : \text{PosVar}(P) \rightarrow \mathcal{T}(\Sigma) \mid \forall i \in \text{PosVar}(P) : I(i) \in \text{Terms}(h, N(i)) \wedge \forall i, j \in \text{PosVar}(P) : (I(i) = I(j) \Leftrightarrow i \equiv_P j)\}$
- $\text{Terms}(h, q) = \{f(I(1), \dots, I(m)) \mid (f(q_1, \dots, q_m) \xrightarrow{c} q) \in \Delta \wedge m > 0 \wedge I \in \text{Sols}(h - 1, c, \{1 \mapsto q_1, \dots, m \mapsto q_m\})\}$

As we have mentioned above, we are not interested in computing all the reached terms at a certain height h , but only how many of them reach each state, up to a certain bound: the maximum arity of the function symbols in Σ .

Definition 13. *Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a $\text{dcTACBB}_{\text{nHe}}^F$. Let h be a natural number. Let q be a state in Q . Then, $\#\text{Terms}_A(h, q)$ is defined as $\min(\text{maxar}, |\text{Terms}_A(h, q)|)$. Let c be a normalized height and equality constraint. Let N be a partial mapping from natural numbers to Q such that $\text{Dom}(N) \supseteq \text{PosVar}(c)$. Then, $\#\text{Sols}_A(h, c, N)$ and $\#\text{AccSols}_A(h, c, N)$ are respectively defined as $\min(\text{maxar}, |\text{Sols}_A(h, c, N)|)$ and $\min(\text{maxar}, |\text{AccSols}_A(h, c, N)|)$. When A is clear from the context, we omit the subindex of $\#\text{Terms}_A$, $\#\text{Sols}_A$, $\#\text{AccSols}_A$.*

The configuration of A for height h is the description of all values $\#\text{Sols}_A(h, c, N)$, $\#\text{AccSols}_A(h, c, N)$, $\#\text{Terms}_A(h, q)$ for $\langle c, N \rangle \in C_A$ and $q \in Q$.

The following recursive definition follows from Definition 13 and Lemma 9.

Lemma 10. Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a $\text{dcTACBB}_{\text{nHe}}^F$. $\#Sols$, $\#AccSols$ and $\#Terms$ can also be defined recursively as follows, where $B(P, N, q)$ denotes $|\{P' \in P \mid N(P') = \{q\}\}|$:

- If $h < 0$, then $\#Terms(h, q) = \#Sols(h, c, N) = \#AccSols(h, c, N) = 0$ for any q, c, N .
- Consider the case $h = 0$. Then, $\#Terms(h, q)$ is defined as $\min(\text{maxar}, |\{a \in \Sigma^{(0)} \mid A(a) = q\}|)$. Moreover, if c is not just a partition, then $\#Sols(h, c, N) = \#AccSols(h, c, N) = 0$, and otherwise if c is a partition P , then $\#Sols(h, c, N)$ and $\#AccSols(h, c, N)$ are defined as $\min(\text{maxar}, \prod_{q \in N(\text{PosVar}(P))} (\frac{\#Terms(h, q)}{B(P, N, q)}) \cdot B(P, N, q)!)$. In the rest of cases assume $h > 0$.
- $\#AccSols(h, c, N) = \min(\text{maxar}, \#AccSols(h - 1, c, N) + \#Sols(h, c, N))$
- $\#Sols(h, c' =_{h'} P, N) = \min(\text{maxar}, \#Sols(h - h', c', N) \cdot \#Sols(h, P, N))$
- $\#Sols(h, c' \leq_{h'} P, N) = \min(\text{maxar}, \#AccSols(h - h', c', N) \cdot \#Sols(h, P, N))$
- $\#Sols(h, P, N) = \min(\text{maxar}, \prod_{q \in N(\text{PosVar}(P))} (\frac{\#Terms(h, q)}{B(P, N, q)}) \cdot B(P, N, q)!)$
- $\#Terms(h, q) = \min(\text{maxar}, \sum_{(f(q_1, \dots, q_m) \xrightarrow{c} q) \in \Delta, m > 0} \#Sols(h - 1, c, \{1 \mapsto q_1, \dots, m \mapsto q_m\}))$

Corollary 6. Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a $\text{dcTACBB}_{\text{nHe}}^F$ having width w . Let h be a natural number. The configuration of A for height $h + w$ can be obtained from the configurations of A for heights $h, h + 1, \dots, h + w - 1$ with time in $\mathcal{O}((\text{maxar}^2 \cdot |\Delta| + |Q|) \cdot (\log(\text{maxar}) + \log(w)))$.

The previous result leads to the fact that, when we find the same w consecutive configurations starting at two different heights $h_1 < h_2$, the sequence of configurations is periodic with period $h_2 - h_1$ starting from h_1 . Thus, deciding emptiness corresponds to check whether there exists an accepting state q satisfying $\#Terms(h, q) \geq 1$ for $h < h_2$, and deciding finiteness corresponds to check whether there exists an accepting state q satisfying $\#Terms(h, q) \geq 1$ for $h_1 \leq h < h_2$. The time complexity depends on Corollary 6 and the number of different possible groups of w consecutive configurations.

Lemma 11. Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a $\text{dcTACBB}_{\text{nHe}}^F$. The number of different configurations of A considering all possible h is bounded by $(\text{maxar} + 1)^{|Q| + |C_A|} \cdot (|C_A| \cdot \text{maxar} + 1)$, i.e., by $(\text{maxar} + 1)^{|Q| + 2 \cdot \text{maxar} \cdot |\Delta|} \cdot (2 \cdot \text{maxar}^2 \cdot |\Delta| + 1)$.

Proof. Similar to Lemma 3, except that instead of dealing with truth values we have numbers in $\{0, \dots, \text{maxar}\}$, and that we need to take into account the number $(\text{maxar} + 1)^{|C_A|}$ of different $\#Sols$.

Corollary 7. Emptiness and finiteness of the language recognized by a $\text{dcTACBB}_{\text{nHe}}^F$ $A = \langle Q, \Sigma, F, \Delta \rangle$ having width w are decidable with time in $2^{\mathcal{O}(w \cdot \log(\text{maxar}) \cdot (|Q| + \text{maxar} \cdot |\Delta|))}$.

As a consequence of Lemma 7 we also obtain decidability for TACBB_{He} .

Corollary 8. *Emptiness and finiteness of the language recognized by a TACBB_{He} $A = \langle Q, \Sigma, F, \Delta \rangle$ are decidable with time in $2^{2^{\mathcal{O}(\log(|\Sigma|) + \text{maxar} \cdot (|Q| + \text{maxar} + \log(w)) + \log(|A|))}}$, where w is the width of A .*

As a final remark, we consider the simpler cases of TACBB_{He} and TACBB_{H} . For the normalization A' of a TACBB_{He} A , since the normalized constraints would only have operators of the form $\leq_1$, we could refine Lemma 11 as we did in Lemma 3 and ignore the number of different $\#Sols$, thus obtaining $(\text{maxar} + 1)^{|Q'|} \cdot (2 \cdot \text{maxar}^2 \cdot |\Delta'| + 1)$ as the number of possible configurations of A' . Moreover, for the sequence of configurations to become periodic it would suffice that two configurations coincided (since we look for identical groups of $w = 1$ consecutive configurations), and hence we would get the following result.

Corollary 9. *Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a TACBB_{He} . Emptiness and finiteness of $\mathcal{L}(A)$ are decidable with time in $2^{\mathcal{O}(\log(\text{maxar}) \cdot 2^{|Q|} + \log(|\Sigma|) + \text{maxar} \cdot (|Q| + \text{maxar}) + \log(|A|))}$.*

For the normalization A' of a TACBB_{H} A , it is not required to obtain a deterministic and complete automaton, and instead of partitions we deal with sets, as for TACBB_{H} . Thus, a refined normalization process can preserve the same state set and generate the rule set Δ' combining ideas from Lemmas 1 and 7 with time in $2^{\mathcal{O}(\text{maxar}^2 + \text{maxar} \cdot \log(w) + \log(|A|))}$. Also, we do not need to count the number of terms for each state, just a truth value. Hence, the cost of computing a new configuration of A' (refinement of Corollary 6) would be in $\mathcal{O}((\text{maxar}^2 \cdot |\Delta'| + |Q|) \cdot \log(w))$, and the number of configurations of A' (refinement of Lemma 11) would be bounded by $2^{|Q| + 2 \cdot \text{maxar} \cdot |\Delta'|} \cdot (2 \cdot \text{maxar} \cdot |\Delta'| + 1)$.

Corollary 10. *Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a TACBB_{H} having width w . Emptiness and finiteness of $\mathcal{L}(A)$ are decidable with time in $2^{2^{\mathcal{O}(\text{maxar}^2 + \text{maxar} \cdot \log(w) + \log(|A|))}}$.*

6 Conclusion

We have defined a new kind of automata with constraints between brothers, extending the current literature by allowing constraints that test the height of subterms. The obtained time complexity for deciding emptiness and finiteness is exponential for the simplest case TACBB_{H} , and double exponential for TACBB_{H} , TACBB_{He} , and TACBB_{He} . For TACBB_{He} and TACBB_{He} both problems are at least EXPTIME-hard due to the equality tests, but the precise hardness for our automata is unknown and deserves further analysis. It would also be interesting to study other extensions of the constraints. In particular, we have focused on constraints between brother positions, but emptiness and finiteness with arbitrary positions for the height constraints might also be decidable. Moreover, several classes of (dis)equality constraints with non-brother positions are known to be decidable, such as reduction automata [10] or $\text{TA}_{\text{hom}, \approx}$ [15]. Extending those models with height constraints might preserve the decidability, and should also be considered.

References

1. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press, New York (1998)
2. Barguñó, L., Creus, C., Godoy, G., Jacquemard, F., Vacher, C.: Decidable classes of tree automata mixing local and global constraints modulo flat theories. *Logical Methods in Computer Science* 9(2) (2013)
3. Bogaert, B., Tison, S.: Equality and Disequality Constraints on Direct Subterms in Tree Automata. In: Symposium on Theoretical Aspects of Computer Science (STACS). pp. 161–171 (1992)
4. Caron, A.C., Seynhaeve, F., Tison, S., Tommasi, M.: Deciding the satisfiability of quantifier free formulae on one-step rewriting. In: Rewriting Techniques and Applications (RTA). pp. 103–117 (1999)
5. Comon, H., Jacquemard, F.: Ground reducibility is EXPTIME-complete. *Information and Computation* 187(1), 123–153 (2003)
6. Comon, H., Cortier, V.: Tree automata with one memory, set constraints and cryptographic protocols. *Theoretical Computer Science* 331(1), 143–214 (2005)
7. Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Löding, C., Lugiez, D., Tison, S., Tommasi, M.: Tree Automata Techniques and Applications. <http://tata.gforge.inria.fr> (2007)
8. Comon-Lundh, H., Jacquemard, F., Perrin, N.: Visibly tree automata with memory and constraints. *Logical Methods in Computer Science* 4(2:8) (2008)
9. Creus, C., Gascón, A., Godoy, G., Ramos, L.: The HOM problem is EXPTIME-complete. In: Logic in Computer Science (LICS). pp. 255–264 (2012)
10. Dauchet, M., Caron, A.C., Coquidé, J.L.: Automata for reduction properties solving. *Journal of Symbolic Computation* 20(2), 215–233 (1995)
11. Doner, J.: Tree acceptors and some of their applications. *Journal of Computer System Sciences* 4, 406–451 (1970)
12. Filiot, E., Talbot, J., Tison, S.: Tree automata with global constraints. *International Journal of Foundations of Computer Science* 21(4), 571–596 (2010)
13. Gécseg, F., Steinby, M.: Tree Automata. *Akadémiai Kiadó* (1984)
14. Gécseg, F., Steinby, M.: Tree languages. In: Rozenberg, G., Salomaa, A. (eds.) *Handbook of Formal Languages*, vol. 3, pp. 1–68. Springer-Verlag (1997)
15. Godoy, G., Giménez, O.: The HOM problem is decidable. *Journal of the ACM* 60(4), 23 (2013)
16. Jacquemard, F., Rusinowitch, M., Vigneron, L.: Tree automata with equality constraints modulo equational theories. *Journal of Logic and Algebraic Programming* 75(2), 182–208 (2008)
17. Mezei, J., Wright, J.B.: Algebraic automata and context-free sets. *Information and Control* 11, 3–29 (1967)
18. Mongy, J.: Transformation de noyaux reconnaissables d’arbres. Forêts RATEG. Ph.D. thesis, Laboratoire d’Informatique Fondamentale de Lille, Université des Sciences et Technologies de Lille, Villeneuve d’Ascq, France (1981)
19. Murata, M., Lee, D., Mani, M., Kawaguchi, K.: Taxonomy of XML schema languages using formal language theory. *ACM Transactions of Internet Technologies* 5(4), 660–704 (2005)
20. Treinen, R.: Predicate logic and tree automata with tests. In: Tiuryn, J. (ed.) *Proc. of the 3rd Int. Conf. on Foundations of Software Science and Computation Structures, FoSSaCS. LNCS*, vol. 1784, pp. 329–343. Springer (2000)