On Positive TAGED with a Bounded Number of Constraints

Pierre-Cyrille Héam[⋆], Vincent Hugot[⋆⋆], and Olga Kouchnarenko

FEMTO-ST CNRS 6174, University of Franche-Comté & INRIA/CASSIS, France
{pierre-cyrille.heam,vincent.hugot,olga.kouchnarenko}@inria.fr

Abstract. Tree Automata With Global Equality Constraints (aka. positive TAGED, or TAGE) are a variety of Bottom-Up Tree Automata, with added expressive power. While there is interest in using this formalism to extend existing regular model-checking frameworks – built on vanilla tree automata – such a project can only be practical if the algorithmic complexity of common decision problems is kept tractable. Unfortunately, useful TAGE decision problems sport very high complexities: Membership is NP-complete, Emptiness and Finiteness are both ExpTime-complete, Universality and Inclusion are undecidable. It is well-known that restricting the *kind* of equality constraints can have a dramatic effect on complexity, as evidenced by Rigid Tree Automata. However, the influence of the *number* of constraints on complexity has yet to be examined. In this paper, we focus on three common decision problems: Emptiness, Finiteness and Membership, and study their algorithmic complexity under a bounded number of equality constraints.

1 Introduction

Tree Automata are a pervasive tool of contemporary Computer Science, with applications running the gamut from XML processing [8] to program verification. Since their original introduction in the fifties, they have spawned an ever-growing family of variants, each with its own characteristics of expressiveness and decision complexity. Notable among them is the sub-family of Tree Automata With Constraints, which increases the expressiveness of vanilla tree automata by providing some means of comparing subtrees. Examples of such devices are are Automata With Equality and Disequality Constraints [4], Automata with Constraints on Brothers [2], and Visibly Tree Automata with Memory and Constraints [3]. In this paper, we focus on one of the latest strains: Tree Automata With Global Equality Constraints (TAGE) [6,5]. Their increased expressiveness is well paid for in terms of algorithmic complexity: Membership is NP-complete, Emptiness and Finiteness are both ExpTime-complete [1], Universality and Inclusion are undecidable. While those complexities are fairly prohibitive, restrictions on the

[⋆] This author is supported by the project ANR 2010 BLAN 0202 02 FREC.
[⋆⋆] This author is supported by the French DGA (Direction Générale de l'Armement).

constraints can dramatically simplify some problems — for instance Rigid Tree Automata (`RTA`) [9], a more restrictive class of `TAGE`, enjoy a trivial, linear-time decision procedure for Emptiness. An application of `TAGE` of particular interest is the extension of regular model-checking techniques, where the increased expressiveness permits a wider range of applications. For such extensions to be practical, algorithmic complexities must be kept tractable — for instance `RTA` achieve that for Emptiness by restricting the *kind* of equality constraints which may be taken; in contrast, the present paper studies how bounding the *number* of constraints influences the complexity of three common decision problems: Emptiness and Finiteness (Sec. 3) are shown to be in PTime for one constraint, and ExpTime-complete for two or more; Membership (Sec. 4) is shown to stay in PTime, regardless of how high the bound is.

2 Preliminaries

Relations & Intervals. Let $R \subseteq Q^2$ be a binary relation on a set Q; we denote by R^+, R^* and $R^\equiv$ its transitive, reflexive-transitive, and equivalence closure (symmetric-reflexive-transitive), respectively. Unless explicitly stated otherwise, reflexive closures are taken on $\operatorname{dom}(R) = \{\, x \mid \exists y : xRy \text{ or } yRx \,\}$, even if R has been introduced as a relation on the larger set Q. The integer interval $[n, m] \cap \mathbb{Z}$ is written $[\![n, m]\!]$.

Trees. We denote by $\mathbb{N}^*$ the set of words over $\mathbb{N}$; if $v, w \in \mathbb{N}^*$, then $v.w$ stands for the concatenation of the words v and w. A *ranked alphabet* is a finite set Σ of symbols, equipped with an arity function $\mathsf{arity} : \Sigma \to \mathbb{N}$. The subset of symbols of Σ with arity n is denoted by Σ_n, and the notation $\sigma/_n$ is shorthand for "σ, with $\mathsf{arity}\,\sigma = n$". The set $\mathcal{T}(\Sigma)$ of trees over Σ is defined inductively as the smallest set such that $\Sigma_0 \subseteq \mathcal{T}(\Sigma)$ and, if $n \geqslant 1$, $\sigma \in \Sigma_n$ and $u_1, \dots, u_n \in \mathcal{T}(\Sigma)$, then $\sigma(u_1, \dots, u_n) \in \mathcal{T}(\Sigma)$. If t is a tree, then the set of *positions* (or *nodes*) $\mathcal{P}os(t) \subseteq \mathbb{N}^*$ is defined inductively by $\mathcal{P}os(t) = \{\varepsilon\}$ if $t \in \Sigma_0$ and $\mathcal{P}os\big(\sigma(u_1, \dots, u_n)\big) = \{\varepsilon\} \cup \{\, k.\alpha_k \mid k \in [\![1, n]\!]\,,\ \alpha_k \in \mathcal{P}os(u_{k+1}) \,\}$ otherwise, where n is the arity of σ. We see a tree t as a function $t : \mathcal{P}os(t) \to \Sigma$ which maps a position to the symbol at that position in t. Positions are equipped with a non-strict (resp. strict) partial order $\trianglelefteq$ (resp. $\triangleleft$), such that $\alpha \trianglelefteq \beta$ iff β is a prefix of α (resp. $\alpha \trianglelefteq \beta$ and $\alpha \neq \beta$). The *subtree of a tree* t *at position* $\alpha \in \mathcal{P}os(t)$ is the tree $t|_\alpha$ such that $\mathcal{P}os(t|_\alpha) = \{\, \beta \mid \alpha.\beta \in \mathcal{P}os(t) \,\}$ and $\forall \beta \in \mathcal{P}os(t|_\alpha)$, $t|_\alpha(\beta) = t(\alpha.\beta)$. Subterms are ordered by the relations $u \trianglelefteq v \iff \exists \alpha \in \mathcal{P}os(v) : v|_\alpha = u$ and $u \triangleleft v \iff u \trianglelefteq v \wedge u \neq v$. Note that $\alpha \trianglelefteq \beta \implies t|_\alpha \trianglelefteq t|_\beta$. Two positions α and β are *incomparable*, written $\alpha \curlywedge \beta$, if neither $\alpha \trianglelefteq \beta$ nor $\beta \trianglelefteq \alpha$. The *size* of a tree t is denoted by $\|t\|$ and defined by $\|t\| = |\mathcal{P}os(t)|$.

Tree Automata. Let Q be a finite set of symbols of arity 0, called *states*, such that $Q \cap \Sigma = \emptyset$. A *transition* is a rewrite rule $\sigma(q_1, \dots, q_n) \to q$, where $q_1, \dots, q_n, q \in Q$ and $\sigma \in \Sigma_n$. A *bottom-up non-deterministic finite tree automaton* (tree automaton, or `TA` for short) over Σ is a tuple $\mathcal{A} = \langle \Sigma, Q, F, \Delta \rangle$, such that $F \subseteq Q$ and Δ is a finite set of transitions. A *run* of $\mathcal{A}$ on a term $t \in \mathcal{T}(\Sigma)$ is a tree $\rho : \mathcal{P}os(t) \to Q$ such that for all $\alpha \in \mathcal{P}os(t)$, $t(\alpha)(\rho(\alpha.1), \dots, \rho(\alpha.n)) \to \rho(\alpha) \in \Delta$. A run ρ is a q-run if $\rho(\varepsilon) = q$, and it is called *accepting* (or *successful*) if $\rho(\varepsilon) \in F$. A set of trees is called a *language*. The set of all trees on which there exists a q-run of $\mathcal{A}$ is written $\mathcal{L}^q(\mathcal{A})$, and the set of trees on which there exists an accepting run is denoted by $\mathcal{L}(\mathcal{A}) = \bigcup_{q \in F} \mathcal{L}^q(\mathcal{A})$, and called the *language recognised* (or *accepted*) by $\mathcal{A}$.

Tree Automata with Equality Constraints: TAGE. A `TAGE`, or "positive `TAGED`" [6] is a tuple $\mathcal{A} = \langle \Sigma, Q, F, \Delta, \cong \rangle$, where $\langle \Sigma, Q, F, \Delta \rangle$ is a tree automaton over Σ and $\cong \subseteq Q^2$ is a binary relation on Q. The *underlying tree automaton* $\langle \Sigma, Q, F, \Delta \rangle$ is denoted by $\mathsf{ta}(\mathcal{A})$. A run of a `TAGE` $\mathcal{A}$ on a tree t is a run of $\mathsf{ta}(\mathcal{A})$ on t *satisfying* the equality constraints of $\cong$, which is to say: for all positions $\alpha, \beta \in \mathcal{P}os(t)$, if $\rho(\alpha) \cong \rho(\beta)$ then $t|_\alpha = t|_\beta$. An *accepting* run of $\mathcal{A}$ is a run of $\mathcal{A}$ which is accepting for $\mathsf{ta}(\mathcal{A})$; accepted languages are defined similarly to `TA`. The Membership problem for `TAGE` is NP-complete [6]. Emptiness and Finiteness are ExpTime-complete, whereas Universality and Inclusion are undecidable [9, Table 1]. Following the respective definitions of runs, it is straightforward that for every `TAGE` $\mathcal{A}$, $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathsf{ta}(\mathcal{A}))$. A `TAGE` $\mathcal{A}$ is said to be *rigid* (i.e. a `RTA`) if $\cong \subseteq id_Q$, i.e. if every constraint is of the form $p \cong p$. The standard *disjoint union* of two `TAGE` $\mathcal{A}$ and $\mathcal{B}$ is a `TAGE` $\mathcal{A} \uplus \mathcal{B}$, such that $\mathcal{L}(\mathcal{A} \uplus \mathcal{B}) = \mathcal{L}(\mathcal{A}) \cup \mathcal{L}(\mathcal{B})$ [6]. Two `TAGE` $\mathcal{A}$ and $\mathcal{B}$ are said to be *equivalent* if $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B})$.

TAGE-Specific Notations. Throughout this paper, any `TAGE` $\mathcal{X}$ will be assumed to have attributes of the form $\langle \mathcal{X}{:}\Sigma, \mathcal{X}{:}Q, \mathcal{X}{:}F, \mathcal{X}{:}\Delta, \mathcal{X}{:}\cong \rangle$. In addition, $\mathcal{A}$ will simply be assumed to be $\langle \Sigma, Q, F, \Delta, \cong \rangle$. We write the modification of an existing `TAGE` as $\wr \mathcal{X} \mid$ <*modifs*>$\int$, where <*modifs*> is a comma-separated list of attribute changes. For brevity, within the scope of $\wr \mathcal{X} \mid \cdots \int$ any unqualified attribute x stands for $\mathcal{X}{:}x$ — this takes precedence over the $\mathcal{A}{:}x$ convention. For instance, $\wr \mathcal{X} \mid \cong := \emptyset \int$ is the bare tree automaton associated with $\mathcal{X}$, or $\mathsf{ta}(\mathcal{X})$. Modifications of the form "$x := f(x)$" will just be written "$f(x)$"; for instance $\wr \mathcal{X} \mid Q \setminus \{q\} \int$ is $\mathcal{X}$ from which the state q has been removed, as with "$Q := Q \setminus \{q\}$" (or even "$\mathcal{X}{:}Q := \mathcal{X}{:}Q \setminus \{q\}$"). Of course in this example the modification "$F \setminus \{q\}$" is completely omitted, as it is implied by “$Q \setminus \{q\}$”, given that by definition $\mathcal{X}{:}F \subseteq \mathcal{X}{:}Q$. The same goes for the removal of all the rules of $\mathcal{X}{:}\Delta$ and constraints of $\mathcal{X}{:}\cong$ that used q.

Tree Automata With Bounded Equality Constraints: $\mathtt{TAGE}_k$. A $\mathtt{TAGE}_k$, where $k \in \mathbb{N}$, is a $\mathtt{TAGE}$ whose number of constraints is at most k. In other words, a $\mathtt{TAGE}_k$ $\mathcal{A}$ is such that $\operatorname{Card}(\cong) \leqslant k$. By extension, we also denote by $\mathtt{TAGE}_k$ the set of all automata which are $\mathtt{TAGE}_k$. Note that trivially $\mathtt{TAGE}_k \subseteq \mathtt{TAGE}_{k+1} \subseteq \mathtt{TAGE}$.

3 The Emptiness & Finiteness Problems

Lemma 1 (Incomparable Positions)**.** *Let $\mathcal{A}$ be a $\mathtt{TAGE}$ with the constraint $p \cong q$, and ρ an accepting run of $\mathcal{A}$ on a tree t. Assume that both those states are involved in the run: $\{p, q\} \subseteq \operatorname{ran} \rho$; then any two distinct positions $\alpha, \beta \in \rho^{-1}(\{p, q\})$, $\alpha \neq \beta$, are incomparable: $\alpha \curlywedge \beta$.*

Proof. Since $\alpha, \beta \in \rho^{-1}(\{p, q\})$ and $\{p, q\} \subseteq \operatorname{ran} \rho$ and $p \cong q$, we have $t|_\alpha = t|_\beta$ Suppose wlog. that $\alpha \lhd \beta$, then $t|_\alpha \lhd t|_\beta$; this is absurd since $t|_\beta$ cannot be structurally equal to one of its own strict subterms. Therefore $\alpha \curlywedge \beta$. □

Lemma 2 (Rigidification)**.** *For every $\mathtt{TAGE}_1$ $\mathcal{A}$, there exists an equivalent $\mathtt{RTA}$ $\mathcal{B}$ whose size is at most quadratic in that of $\mathcal{A}$.*[a]

Proof. If $\mathcal{A}$ has no constraints, or a rigid constraint ($p \cong p$), then $\mathcal{B} = \mathcal{A}$. Assume $\mathcal{A}$ has a constraint of the form $p \cong q$, with $p \neq q$, and suppose wlog. that $p, q \notin F$. Building Blocks. We let $\mathcal{B}_p^\neg = \wr \mathcal{A} \mid Q \setminus \{p\} \int$, $\mathcal{B}_q^\neg = \wr \mathcal{A} \mid Q \setminus \{q\} \int$, $\mathcal{B}_p = \wr \mathcal{B}_q^\neg \mid F := \{p\}, \Delta := \Delta_p \int$ —where Δ_p is $\mathcal{B}_q^\neg{:}\Delta$ from which all rules where p appears in the left-hand side have been removed, and $\mathcal{B}_q$, which is defined symmetrically to $\mathcal{B}_p$. Lastly, $\mathcal{B}_{pq}$ is built to accept the intersection of the languages of $\mathcal{B}_p$ and $\mathcal{B}_q$; using the standard product algorithm, it has a single final state $q_f = (p, q)$. Note that they are all vanilla tree automata. Construction. We let

$$\mathcal{B} = \mathcal{B}_p^\neg \uplus \mathcal{B}_q^\neg \uplus \wr \mathcal{A} \mid Q', \Delta', q_f \cong q_f \int, \text{ with } \begin{cases} Q' = (Q \setminus \{p, q\}) \uplus (\mathcal{B}_{pq}{:}Q) \\ \Delta' = \Delta_{pq}^{q_f} \uplus (\mathcal{B}_{pq}{:}\Delta) \end{cases},$$

where $\Delta_{pq}^{q_f}$ is $\mathcal{A}{:}\Delta$ from which all left-hand side occurrences of p or q have been replaced by q_f. Equivalence. Let $t \in \mathcal{L}(\mathcal{A})$, accepted through a run ρ; one of the following is true: **(1)** neither p nor q appears in ρ **(2)** p appears, and q does not **(3)** q appears, and p does not **(4)** both p and q appear. In the three first cases, the

[a] Note that the general construction for $\mathtt{TAGE}$ is exponential [6, Thm. 10].

constraints are not involved, and t is accepted by: (1) both $\mathcal{B}_p^-$ and $\mathcal{B}_q^-$ (2) $\mathcal{B}_q^-$ (3) $\mathcal{B}_p^-$. In case (4), a subterm evaluating to p will belong to $\mathcal{L}^p(\mathcal{A})$ by definition, and also to $\mathcal{L}^q(\mathcal{A})$ as it needs to be equal to another extant subterm evaluating to q. Furthermore, p and q can only appear at the root of each subruns, lest $p \approxeq q$ be trivially violated. Therefore, a successful run of $\mathcal{B}$ can be constructed by simply substituting all p and q subruns by q_f-runs of $\mathcal{B}_{pq}$. Thus $t \in \mathcal{L}(\mathcal{B})$. Conversely, let $t \in \mathcal{L}(\mathcal{B})$; it is immediately seen by construction that $\mathcal{L}(\mathcal{B}_p^-) \subseteq \mathcal{L}(\mathcal{A})$ and $\mathcal{L}(\mathcal{B}_q^-) \subseteq \mathcal{L}(\mathcal{A})$. Suppose that t is accepted through a run of the third and last part of $\mathcal{B}$ (namely $\wr \mathcal{A} \mid \cdots \int$); then every q_f-subrun can be replaced by either a p-run or a q-run of $\mathcal{A}$. The result of this operation is trivially an accepting run of $\mathsf{ta}(\mathcal{A})$; there remains to observe that it satisfies $p \approxeq q$, because the corresponding subtrees must be equal given the constraint $(q_f, q_f) \in \mathcal{B}{:}\approxeq$. Thus $t \in \mathcal{L}(\mathcal{A})$. Size & Time. All building blocks are of size $O(\|\mathcal{A}\|)$, except $\mathcal{B}_{pq}$, which is of size $O(\|\mathcal{A}\|^2)$. Globally, the size of $\mathcal{B}$ is at most quadratic in that of $\mathcal{A}$. The construction is also straightforwardly done in quadratic time. □

***Proposition 3** (Emptiness).* *The Emptiness problem is in* PTime *for* TAGE$_1$*, and* ExpTime*-complete for* TAGE$_2$*.*

Proof. **TAGE$_1$.** Emptiness is testable in linear time for RTA [9], therefore the emptiness of $\mathcal{A}$ is testable in quadratic time using the construction of Lemma 2. **TAGE$_2$.** Overview. We reduce the test of the emptiness of the intersection of n tree automata $\mathcal{A}_1, \dots, \mathcal{A}_n$, which is an ExpTime-complete problem, to the emptiness of a TAGE$_2$ $\mathcal{A}$. This is similar to the arguments of [5, Thm. 1], the major difference being that we can only use two constraints instead of an unbounded number of constraints. The idea is to take advantage of the fact that an explicit equality constraint between two positions effectively enforces an arbitrary number of implicit equality constraints on the sub-positions. Assumptions. It is assumed wlog. that $n \geqslant 2$ and the sets of states of the $\mathcal{A}_i$ are pairwise disjoint; that is to say, $\forall i, j \in [\![1, n]\!]\,,\ i \neq j \Rightarrow (\mathcal{A}_i{:}Q) \cap (\mathcal{A}_j{:}Q) = \varnothing$. Furthermore, it can be assumed that each $\mathcal{A}_i$ has exactly one final state q_{f_i}. If that is not the case, then $\mathcal{A}_i$ can be modified to be so, which results in its size doubling in the worst case. Language. We define the language L as the set of trees of the form given in Figure 1[p6], where σ is a fresh binary symbol and for all i, $x_i \in \mathcal{L}(\mathcal{A}_i)$ and $x = x_i$. Note that this implies that $x \in \bigcap_i \mathcal{L}(\mathcal{A}_i)$, and therefore L is empty iff $\bigcap_i \mathcal{L}(\mathcal{A}_i)$ is empty. Automaton. We build a TAGE$_2$ $\mathcal{A}$ that accepts L, by first building a universal tree automaton $\mathcal{U}$, of final state q^u. Then, we let $\mathcal{A} = \langle \Sigma, Q, F, \Delta, \approxeq \rangle$, where

$$Q = (\biguplus_i \mathcal{A}_i{:}Q) \uplus (\mathcal{U}{:}Q) \uplus \{\, q_1^u, \dots, q_{n-1}^u, q_1^v, \dots, q_{n-1}^v \,\} \uplus \{q_f\}$$

$$\begin{tikzcd}
 & \sigma \arrow[r] \arrow[d] & \sigma \arrow[r] \arrow[d] & \sigma \arrow[r] \arrow[d] & \cdots \arrow[r] & \sigma \arrow[r] \arrow[d] & x_n \\
\sigma \arrow[ur] \arrow[dr] & x_1 & x_2 & x_3 & & x_{n-1} & \\
 & \sigma \arrow[r] \arrow[d] & \sigma \arrow[r] \arrow[d] & \sigma \arrow[r] \arrow[d] & \cdots \arrow[r] & \sigma \arrow[r] \arrow[d] & x \\
 & x & x & x & & x &
\end{tikzcd}$$

Fig. 1: Language L

$$\begin{aligned}
F = \{q_f\} \qquad & q^u \approxeq q^u, \quad q_1^u \approxeq q_1^v \qquad \Sigma = \left(\textstyle\bigcup_i \mathcal{A}_i : \Sigma\right) \uplus \{\sigma/_2\} \\
\Delta = & \{\, \sigma(q_1^u, q_1^v) \to q_f \,\} \cup \left(\textstyle\bigcup_i \mathcal{A}_i : \Delta\right) \cup (\mathcal{U} : \Delta) \cup \\
& \left\{\, \sigma(q^u, q_{k+1}^u) \to q_k^u \;\middle|\; k \in [\![1, n-2]\!] \,\right\} \cup \left\{\, \sigma(q^u, q^u) \to q_{n-1}^u \,\right\} \cup \\
& \left\{\, \sigma(q_{f_k}, q_{k+1}^v) \to q_k^v \;\middle|\; k \in [\![1, n-2]\!] \,\right\} \cup \left\{\, \sigma(q_{f_{n-1}}, q_{f_n}) \to q_{n-1}^v \,\right\} \ .
\end{aligned}$$

Note that we have $\mathcal{L}(\mathcal{A}) = L$ and $\|\mathcal{A}\| = O\left(\sum_{k=1}^{n} \|\mathcal{A}_i\|\right)$, which concludes the proof. □

Proposition 4 (*Finiteness*)**.** *The Finiteness problem is in* PTime *for* TAGE$_1$*, and* ExpTime*-complete for* TAGE$_2$*.*

Proof. **TAGE$_1$.** Finiteness is testable in linear time for RTA [9], therefore the finiteness of $\mathcal{A}$ is testable in quadratic time using the construction of Lemma 2.
TAGE$_2$. We reduce the Emptiness problem for TAGE$_2$ to the Finiteness problem. Given a TAGE$_2$ $\mathcal{A}$, we build

$$\begin{aligned}
\mathcal{A}' = \wr\, \mathcal{A} \mid Q \uplus \{p\}, F := \{p\}, \Sigma \uplus \{\sigma/_1\}, \Delta' \,\wr \\
\text{where } \Delta' = \Delta \cup \{\, \sigma(q_f) \to p \mid q_f \in F \,\} \cup \{\, \sigma(p) \to p \,\} \ .
\end{aligned}$$

$\mathcal{A}'$ is also a TAGE$_2$. If $\mathcal{A}$ accepts the empty language, then so does $\mathcal{A}'$. Conversely, if $t \in \mathcal{L}(\mathcal{A})$, then $\sigma^*(t) \subseteq \mathcal{L}(\mathcal{A}')$, and thus $\mathcal{L}(\mathcal{A}')$ is infinite. Consequently, the language of $\mathcal{A}'$ is finite iff that of $\mathcal{A}$ is empty. This, combined with Prp. 3[p5], shows that TAGE$_2$-Finiteness is ExpTime-hard; since the general problem for TAGE is ExpTime [6, Thm. 14], TAGE$_2$-Finiteness is ExpTime-complete. □

4 The Membership Problem

Let us begin with some general observations and notations. We will need to reason about the relation $\approxeq$; unfortunately, it is not an equivalence relation. For instance, given the constraints $p \approxeq r$ and $r \approxeq q$ it is tempting, but in general wrong, to infer $p \approxeq q$ by transitivity. The crux of the matter here is whether the state r actually appears in the run: if it does, $p \approxeq q$ is effectively implied, but if it does not, then both constraints $p \approxeq r$ and $r \approxeq q$ are moot. Lemma 5 shows that, given the knowledge (or the assumption) of a set $P \subseteq \operatorname{dom} \approxeq$ of the constrained states which are actually present in runs, the constraints of $\approxeq$ are interchangeable with an equivalence relation, which we call the *togetherness* relation.

Lemma 5 (Togetherness)**.** *Let $\mathcal{A}$ be a* TAGE *and $P \subseteq \operatorname{dom} \approxeq$. Then any run ρ such that $(\operatorname{ran} \rho) \cap (\operatorname{dom} \approxeq) = P$ is accepting for $\mathcal{A}$ if and only if it is so for $\mathcal{A}_P = \wr \mathcal{A} \mid \approxeq := \left(\approxeq \cap P^2\right)^{\equiv} \wr$, where the closure is meant under* $\operatorname{dom}(\approxeq \cap P^2)$.

Given a P, we denote by $\asymp_P = \left(\approxeq \cap P^2\right)^{\equiv}$ this equivalence relation, and say that "p *and* q *are together wrt.* P" if $p \asymp_P q$. Its equivalence classes are denoted by $\mathbb{G}_P = {}^{\operatorname{dom}(\approxeq \cap P^2)}/_{\asymp_P}$, and called *groups*. If t is a tree, we write $\sim$ for the *similarity relation on* t, defined on $\mathcal{P}os(t)^2$ such that $\alpha \sim \beta \iff t|_\alpha = t|_\beta$. We denote by $\mathbb{S}_t$ the quotient set ${}^{\mathcal{P}os(t)}/_{\sim}$ of the *similarity classes of* t.

Lemma 6 (Housing Groups)**.** *Let $\mathcal{A}$ be a* TAGE*, $P \subseteq \operatorname{dom} \approxeq$ and ρ a run of $\mathcal{A}$ on a tree t, such that $(\operatorname{ran} \rho) \cap (\operatorname{dom} \approxeq) = P$. Then ρ satisfies the constraints of $\approxeq$ if and only if $\forall G \in \mathbb{G}_P$, $\exists C_G \in \mathbb{S}_t : \rho^{-1}(G) \subseteq C_G$.*

Given the hypothesis of P and given a successful run ρ on t, we call the map $G \mapsto C_G$ a P-*housing of* ρ *in* t, which is said to be *compatible with* ρ, and we denote by $\mathbb{H}^t_P = \mathbb{G}_P \to \mathbb{S}_t$ the set of all possible P-housings on t.

Proposition 7 (Membership)**.** *Given an arbitrary but fixed $n \in \mathbb{N}$, the Membership problem for* TAGE_n *is in* PTime *— albeit with an overhead exponential in n.*

Proof. Let $\mathcal{A}$ be a TAGE_n, and t a tree. The Housing Lemma (6[p7]) has already established that a run ρ of $\mathcal{A}$ on t satisfies $\approxeq$ iff there exists a housing $h \in \mathbb{H}^t_P$ which is compatible with ρ, where $P = (\operatorname{dom} \approxeq) \cap (\operatorname{ran} \rho)$ is the set of constrained states which actually appear in the run. Our strategy to check the membership of t will simply be to try each possible $P \subseteq \operatorname{dom} \approxeq$ successively, by attempting, for each possible housing $h \in \mathbb{H}^t_P$, to craft an accepting run ρ of $\mathsf{ta}(\mathcal{A})$ compatible with h. There are at most 2^{2n} possible P, and given a choice of P, there are $|\mathbb{S}_t|^{|\mathbb{G}_P|} \leqslant \|t\|^{2n}$ P-housings on t, which gives at most $4^n \cdot \|t\|^{2n}$ tests in total. Note that since n is a constant, this remains polynomial. There only remains to show that given a choice of P and $h \in \mathbb{H}^t_P$, the existence of a compatible run can be tested in polynomial time. To do so, we use a variant of the standard reachability algorithm, where only the states of P may appear, and the states of a given group $G \in \mathbb{G}_P$ may only appear at the positions assigned to them by the chosen housing h. Formally, given a choice of P and a housing $h \in \mathbb{H}_P^t$, there exists such a run iff $\Phi_t^{P,h}(\varepsilon) \cap F \neq \emptyset$, where

$$\Phi_t^{P,h}(\alpha) = \left\{ q \in Q \;\middle|\; \begin{array}{l} t(\alpha)(p_1, \dots, p_n) \to q \in \Delta \\ \forall i \in [\![1, n]\!],\ p_i \in \Phi_t^{P,h}(\alpha.i) \\ q \in \bigcup \mathbb{G}_P \implies \alpha \in h([q]_{\asymp_P}) \\ q \notin \mathrm{dom}(\approxeq) \setminus P \end{array} \right\} .$$

The reader will notice that, were the last two conditions removed, $\Phi_t^{P,h}(\alpha)$ would simply be the set of reachable states at position α. The additional two constraints are $O(1)$ operations, thus $\Phi_t^{P,h}(\cdot)$ does run in polynomial time; there only remains to show that our algorithm does what is expected of it. There are two points to this: **(1)** no false negative: every successful run is subsumed by some $\Phi_t^{P,h}(\cdot)$ **(2)** no false positive: every run subsumed by some $\Phi_t^{P,h}(\cdot)$ is accepting.

(1) Let ρ a successful run for $\mathcal{A}$, and $P = (\operatorname{ran} \rho) \cap (\operatorname{dom} \approxeq)$; then by the Housing Lemma, it satisfies $\asymp_P$, and there exists a housing $h \in \mathbb{H}_P^t$ with which it is compatible. We propose that ρ is subsumed by $\Phi_t^{P,h}(\cdot)$, which is to say that for each position $\alpha \in \mathcal{P}os(t)$, we must have $\rho(\alpha) \in \Phi_t^{P,h}(\alpha)$. Indeed, let α any position, and $q = \rho(\alpha)$; we check that q satisfies all four conditions for belonging to $\Phi_t^{P,h}(\alpha)$. The first condition is trivially satisfied since ρ is a run. The second one will be the hypothesis of our recursion which, quite conveniently, evaluates to true vacuously if α is a leaf. The third condition is taken care of by the Housing Lemma: suppose $q \in \bigcup \mathbb{G}_P$; then there is a group $G \in \mathbb{G}_P$ such that $q \in G$ (in fact $G = [q]_{\asymp_P}$), and $\rho^{-1}(G) \subseteq h(G)$. Thus we have the chain $\alpha \in \rho^{-1}(\{q\}) \subseteq \rho^{-1}(G) \subseteq h(G)$, and in particular $\alpha \in h([q]_{\asymp_P})$. The fourth and last condition is trivial given our choice of P: Assume its negation $q \in \mathrm{dom}(\approxeq) \setminus P$, then you have $q \notin \operatorname{ran} \rho$, which is absurd.

(2) Let ρ be a run subsumed by $\Phi_t^{P,h}(\cdot)$, for some P and h. By the fourth condition, $(\operatorname{ran} \rho) \cap (\mathrm{dom}(\approxeq) \setminus P) = \emptyset$, and thus $(\operatorname{ran} \rho) \cap (\operatorname{dom} \approxeq) \subseteq P$. Let $\alpha \in \mathcal{P}os(t)$; by the third condition, if $\rho(\alpha) \in G \in \mathbb{G}_P$, then $\alpha \in h(G)$; in other words, $\rho^{-1}(G) \subseteq h(G)$, thus by the Housing Lemma[(b)], ρ is successful. □

[(b)] The watchful reader will notice that we are slightly cheating here, because Lem. 6[p7] as written requires $(\operatorname{ran} \rho) \cap (\operatorname{dom} \approxeq) = P$. The inclusion is enough for the "if" part, as shown by the relevant halves of the proofs of Lem. 6[p7] and Lem. 5[p7]. Alternatively, one could replace P and h by adequate $P' \subseteq P$ and $h' \in \mathbb{H}_{P'}^t$, such that we have equality and preserve subsumption. Either way this is an easy technicality with no bearing on any other part of this paper.

5 Conclusions

In the case of Emptiness and Finiteness we have shown that, perhaps somewhat counter-intuitively, while the limitation to a single equality constraint does lead to tremendously easier complexities (from ExpTime-hardness to quadratic decision procedures), the addition of a second constraint suffices to reintroduce the full complexity of the general, unbounded problem.

This stands in contrast to the behaviour of the Membership problem which, while NP-complete in general, becomes polynomial once the number of constraints is bounded by a constant, regardless of the size of that constant — though admittedly "polynomial" is in that case quite unlikely to mean "efficient" for anything but the smallest constants. Nevertheless, this suggests a potentially more scalable alternative to the existing SAT encoding approach [7].

Acknowledgements. Our thanks go to the reviewers, especially for the suggestion of a simpler approach to the $\mathtt{TAGE}_1$ parts of Propositions 3 and 4.

References

1. Barguñó, L., Creus, C., Godoy, G., Jacquemard, F., Vacher, C.: The emptiness problem for tree automata with global constraints. In: LICS. pp. 263–272. IEEE Computer Society (2010)
2. Bogaert, B., Tison, S.: Equality and disequality constraints on direct subterms in tree automata. In: STACS. LNCS, vol. 577, pp. 161–171. Springer (1992)
3. Comon-Lundh, H., Jacquemard, F., Perrin, N.: Visibly tree automata with memory and constraints. CoRR abs/0804.3065 (2008)
4. Dauchet, M., Mongy, J.: Transformations de noyaux reconnaissables d'arbres, Forêts RATEG. Ph.D. thesis, LIFL (France) (1981)
5. Filiot, E., Talbot, J., Tison, S.: Tree automata with global constraints. In: Developments in Language Theory. pp. 314–326. Springer (2008)
6. Filiot, E., Talbot, J.M., Tison, S.: Tree automata with global constraints. Int. J. Found. Comput. Sci. 21(4), 571–596 (2010)
7. Héam, P.C., Hugot, V., Kouchnarenko, O.: SAT solvers for queries over tree automata with constraints. In: ICST (CSTVA ws.). pp. 343–348. IEEE (2010)
8. Hosoya, H.: Foundations of XML Processing: The Tree-Automata Approach. Cambridge University Press (2010), http://books.google.fr/books?id=xGlH3ADxwn4C
9. Jacquemard, F., Klay, F., Vacher, C.: Rigid tree automata and applications. Inf. Comput. 209(3), 486–512 (2011)