International Journal of Foundations of Computer Science
© World Scientific Publishing Company

Tree Automata With Global Constraints

Emmanuel Filiot
CS Department, Université Libre de Bruxelles, Bld du Triomphe CP 212
Brussels, 1050, Belgium
efiliot@ulb.ac.be

Jean-Marc Talbot
CS Department, Université de Provence, 39 rue Joliot-Curie
F-13453 Marseille Cedex 13, France
jean-marc.talbot@lif.univ-mrs.fr

Sophie Tison
CS Department, Université des Sciences et Technologies de Lille
59655 Villeneuve d'Ascq CEDEX, France
Sophie.Tison@lifl.fr

Received (31 05 2009)
Revised (2 07 2010)
Accepted (1 08 2010)

We define tree automata with global equality and disequality constraints (TAGED). TAGEDs can test (dis)equalities between subtrees which may be arbitrarily faraway. In particular, they are equipped with an equality relation and a disequality relation on states, so that whenever two subtrees t and t' evaluate (in an accepting run) to two states which are in the (dis)equality relation, they must be (dis)equal. We study several properties of TAGEDs, and prove emptiness decidability of for several expressive subclasses of TAGEDs.

Keywords: Trees, Automata, Equality Constraints.

1991 Mathematics Subject Classification: 68Q45, 68Q68

1. Introduction

The emergence of XML has strengthened the interest in tree automata, as it is a clean and powerful model for XML tree acceptors [20, 21]. In this context, tree automata have been used, for example, to define schemas, and queries, but also to decide tree logics, to type XML transformations, and even to learn queries. However, it is known that sometimes, expressiveness of tree automata is not sufficient. This is the case for instance in the context of non-linear rewriting systems, for which more powerful tree acceptors are needed to decide interesting properties of those rewrite systems. For example, the set of ground instances of $f(x, x)$ is not regular.

Tree automata with constraints have been introduced to overcome this lack of expressiveness [4, 10, 17, 18]. In particular, the transitions of these tree automata are fired as soon as the subtrees of the current tree are (dis)equal. But typically, these constraints are kept local to preserve decidability of emptiness and good closure properties – in particular, tests are performed between siblings or cousins –. In the context of XML, and especially to define tree patterns, one needs global constraints. For instance, one may want to represent the set of ground instances of the pattern $X(\texttt{author}(x), \texttt{author}(x))$, where X is a binary context variable, and x is an author which occur at least twice (we assume this pattern to be matched against XML trees representing a bibliography). In this example, the two subtrees corresponding to the author might be arbitrarily faraway, making the tree equality tests more global. Patterns might be more complex, by the use of negation (which enable to test tree disequalities), Boolean operations, and regular constraints on variables. The TQL logic, in particular, makes possible to define such patterns [5, 12]. In this paper, we introduce *Tree Automata with Global Equalities and Disequalities* (TAGEDs for short), which capture the expressiveness of the guarded TQL fragment with tree variables (assumed to be existentially quantified). These are tree automata A equipped with an equality relation $=_A$ and a disequality relation $\neq_A$ on (a subset of) states. A tree t is accepted by A if there is a computation of A on t which leads to an accepting state, and such that, whenever two subtrees of t evaluate to two states q_1, q_2 (not necessarily equal) such that $q_1 =_A q_2$ (resp. $q_1 \neq_A q_2$), then these two subtrees must be structurally equal (resp. structurally different). TAGEDs may be interesting in their own, since they are somehow orthogonal to usual automata with constraints [4]. Indeed, if we view equality tests during computations as an equivalence relation on a subset of nodes (two nodes being equivalent if their rooted subtrees have been successfully tested to be equal by A), in the former, there are a bounded number of equivalence classes of unbounded cardinality, while in the latter, there might be an unbounded number of equivalence classes of bounded cardinality. Recently, a subclass of TAGEDs has been introduced, namely *rigid tree automata* (RTA) [16]. They are applied to the analysis of security protocols based on the computation of the closure of RTA-languages by different class of term rewriting systems.

In this paper, we study closure properties of TAGEDs and some classical decison problems. In particular, we prove or recall decidability of the emptiness problem for several subclasses, namely *positive TAGEDs* where only equality tests are allowed, *negative TAGEDs* where only disequality tests are allowed, and *vertically bounded TAGEDs*, that perform both kind of tests but a bounded number of disequality tests along every branch. We mainly focus on the proof of emptiness for vertically bounded TAGEDs.

Related Work TAGEDs were first introduced in [12] to decide a fragment of the TQL tree logic. Emptiness for a restriced class of TAGEDs was proved to be decidable, where only a bounded number of equality and disequality tests was

allowed.

It was recently shown in [3] that the emptiness problem for a strictly more expressive class of tree automata with global constraints is decidable. This class, called *tree automata with global constraints* (TAGC), extends TAGEDs with the ability to test disequalities between subtrees which evaluate to the same state. TAGEDs do not allow for such tests as we require that any pair of subtrees which are tested for disequality evaluate to different states. However for TAGCs, the complexity of the emptiness problem is unknown.

Extensions of tree automata which allow for syntactic equality and disequality tests between subterms have already been defined by adding constraints to the rules of automata. E.g., adding the constraint $1.2 = 2$ to a rule means that one can apply the rule at position π only if the subterm at position $\pi.1.2$ is equal to the subterm at position $\pi.2$. Testing emptiness of the recognized language is undecidable in general [19] but two classes with a decidable emptiness problem have been emphasized. In the first class, automata are deterministic and the number of equality tests along a path is bounded [10] whereas the second restricts tests to sibling subterms [4]. This latter class has recently been extended to unranked trees [18], the former one has been extended to equality modulo equational theories [17]. But, contrarily to TAGEDs, in all these classes, tests are performed locally, typically between sibling or cousin subterms.

Automata with local and global equality tests, using one memory, have been considered in [8]. These are tree automata with a memory which can store a single tree. They run in a bottom-up fashion, and the two trees contained in the respective memories coming from two computations on the two subtrees of the current node can be combined thanks to tree operations and compared with tree equality tests. The emptiness problem is decidable for tree automata with memory which can simulate positive TAGEDs (TAGEDs performing only equality tests) which use at most one state per runs to test equalities. However, their ability to perform local tests make them incomparable to TAGEDs.

Rigid tree automata (RTAs) have been introduced in [16]. These are positive TAGEDs with so called *rigid states*. All subtrees that evaluate to the same rigid state have to be equal. These are positive TAGEDs whose equality relation is a subset of the identity relation on states. As shown in Section 4, RTAs are equally expressive as positive TAGEDs.

Finally, automata for DAGs which were studied in [2, 6] cannot be compared to positive TAGEDs, as they run on DAG representations of trees (with maximal sharing). In the other hand, as shown in Example 3, we cannot impose in TAGEDs that every equal subtrees evaluate to the same state in a successful run, since it may be needed to evaluate the leaves to possibly different states.

2. Preliminaries

For the sake of clarity, we consider binary trees, with constant and binary function symbols from a ranked alphabet, but all the definitions can naturally be extended

to ranked trees of arbitrary arity.

Binary trees We start from a ranked alphabet Σ ranged over binary symbols f and constant symbols a. A binary tree $t \in T_\Sigma$ is a finite rooted acyclic directed graph, whose nodes are labeled in Σ. Every node is connected to the root by a unique path. All nodes of binary trees either have 0 or 2 linearly ordered children. Nodes without children are called *leaves*. All other nodes have a distinguished first and second child. For a tree t, ϵ_t denotes its root and N_t its set of nodes. For all nodes $u \in N_t$, we write $t(u)$ its label in Σ. Given two nodes $v_1, v_2 \in N_t$ we call v_2 a *child* of v_1 and write $v_1 \lhd v_2$ if there exists an edge from v_1 to v_2. For all $i \in \{1,2\}$, $v_1.i$ stands for the i-th child of v_1. The *descendant* relation $\lhd^*$ on nodes is the reflexive transitive closure of the child relation $\lhd$. We denote by $\lhd^+$ the strict descendant relation. The *subtree of a tree t rooted by some node $v \in N_t$* is the tree (denoted by $t|_v$) induced by t whose set of nodes is $\{v' \in N_t \mid v \lhd^* v'\}$. It is often convenient to view a tree t as a ground term over Σ. In particular, $f(t_1, t_2)$ denotes a tree whose root is labeled by f and whose first and second subtrees are t_1 and t_2 respectively. Two trees t, t' are *equal*, denoted by $t = t'$, if there is an isomorphism from N_t to $N_{t'}$ which preserves the first and second child relations, and the node labels. Finally, the size of a tree, denoted by $\|t\|$, is its number of nodes.

Path Isomorphism Let $t \in T_\Sigma$, and $u, v \in N_t$ such that $u \lhd^* v$. We denote by $\mathrm{path}_t(u, v)$ the finite sequence of nodes $u_1, \ldots, u_n$ such that $u_1 = u$, $u_n = v$, and for all $i \in \{1, \ldots, n-1\}$, u_{i+1} is a child of u_i. In particular, $\mathrm{path}_t(u, u) = u$. Given two other nodes u', v' such that $u' \lhd^* v'$, we say that $\mathrm{path}_t(u, v)$ is *edge-isomorphic* to $\mathrm{path}_t(u', v')$, denoted by $\mathrm{path}_t(u, v) \equiv \mathrm{path}_t(u', v')$, if v and v' are reachable from u and u' respectively by the same sequence of first-child or second-child edges. More formally, if $\mathrm{path}_t(u, v) = u_1 \ldots u_n$ and $\mathrm{path}_t(u', v') = u'_1 \ldots u'_n$ for some $u_1, \ldots, u_n, u'_1, \ldots, u'_n$, then they are edge-isomorphic if for all $i \in \{1, \ldots, n-1\}$, for all $\alpha \in \{1, 2\}$, $u_i.\alpha = u_{i+1}$ iff $u'_i.\alpha = u'_{i+1}$.

Tree Automata We define tree automata on binary trees, but the reader may refer to [9] for more details. A *tree automaton* is a 4-tuple $A = (\Sigma, Q, F, \Delta)$ where Q is a finite set of states, $F \subseteq Q$ is a set of final states, and Δ is a set of rules of the form $a \to q$ and $f(q_1, q_2) \to q$, where f is a binary function symbol, a a constant, and q_1, q_2, q are states from Q. A *run* of A on a tree t is a tree r over Q such that: (i) $N_r = N_t$, (ii) for all leaves $u \in N_r$, we have $t(u) \to r(u) \in \Delta$, and (iii) for all inner-nodes $u \in N_r$, we have $t(u)(r(u.1), r(u.2)) \to r(u) \in \Delta$. A run r is *successful* if $r(\epsilon) \in F$. The *language defined* by A, denoted $\mathcal{L}(A)$, is the set of trees t for which there exists a successful run of A.

3. TAGEDs : Definition, Examples and Closure Properties

In this section, we define TAGEDs and several subclasses, give examples and study their closure properties as well as some decision problems.

Definition 1. *A TAGED is a 6-tuple $A = (\Sigma, Q, F, \Delta, =_A, \neq_A)$ such that*

(Σ, Q, F, Δ) *is a tree automaton, and* $=_A$, $\neq_A$ *are binary relations on* Q.

A TAGED A is said to be *positive* (resp. negative) if $\neq_A$ (resp. $=_A$) is empty. It is *rigid* if $=_A \subseteq id_Q$ (id_Q is the identity relation on Q). We denote by $\mathrm{dom}(=_A)$ the domain of $=_A$, i.e. $\{q \mid \exists q' \in Q \cdot\ q =_A q' \vee q' =_A q\}$. The set $\mathrm{dom}(\neq_A)$ is defined similarly. A is *reflexive* if $=_A$ is reflexive over $\mathrm{dom}(=_A)$. Note that positive rigid TAGEDs are exactly the class of *rigid tree automata* (RTA) in the terminology of [16]. We denote by TAGED$^+$ (resp. TAGED$^-$) the class of positive (resp. negative) TAGED. The definition of TAGEDs differs from the definition given in [13, 11], as we do not impose any constraint on the two relations $=_A$ and $\neq_A$. However, this new definition does not lead to extra expressive power, as shown by Corollary 11.
The notion of successful run differs from tree automata as we add equality and disequality constraints. Let r be a run of the tree automaton (Σ, Q, F, Δ) on a tree t. An equality (resp. disequality) constraint is a pair of nodes $(u, v) \in {N_t}^2$ such that $r(u) =_A r(v)$ (resp. $r(u) \neq_A r(v)$). The set of equality (resp. disequality) constraints for t and r is denoted by $\mathrm{cst}^A_=(t, r)$ (resp. $\mathrm{cst}^A_\neq(t, r)$). The run r satisfies the equality constraints if $\forall (u, v) \in \mathrm{cst}^A_=(t, r)$, $t|_u = t|_v$. Similarly, it satisfies the disequality constraints if $\forall (u, v) \in \mathrm{cst}^A_\neq(t, r)$, $t|_u \neq t|_v$.

A run is *successful* (or *accepting*) if it is successful for the tree automaton (Σ, Q, F, Δ) and satisfies all the constraints[a]. The language defined by A, denoted $\mathcal{L}(A)$, is the set of trees t having a successful run for A. Two TAGEDs are *equivalent* if they accept the same language. Finally, the size of A, denoted $\|A\|$, is $|Q| + |\Delta| + |\mathrm{dom}(=_A) \cup \mathrm{dom}(\neq_A)|^2$.
As shown by Example 2, TAGEDs are strictly more expressive than tree automata.

Example 2. *Let* $Q = \{q, q_=, q_f\}$, $F = \{q_f\}$, *and* Δ *the following set of rules:*

$$a \to q \quad a \to q_= \quad f(q, q) \to q \quad f(q, q) \to q_= \quad f(q_=, q_=) \to q_f.$$

The language accepted by the TAGED$^+$ $A_1 = (\Sigma, Q, F, \Delta, \{q_= =_{A_1} q_=\})$ *is the set* $\{f(t, t) \mid f \in \Sigma, t \in T_\Sigma\}$, *which is known to be non regular* [9]. *For instance, a successful run of* A_1 *on* $f(f(a, a), f(a, a))$ *is* $q_f(q_=(q, q), q_=(q, q))$.

Example 3. *Let* $\mathcal{X}$ *be a finite set of variables. We now define a TAGED* $A_{sat} = (\Sigma_{sat}, Q_{sat}, F_{sat}, \Delta_{sat})$ *which accepts tree representations of satisfiable Boolean formulas with free variables* $\mathcal{X}$. *The alphabet* Σ_{sat} *consists of the binary symbols* $\wedge, \vee$ *and* x, *for all* $x \in \mathcal{X}$, *the unary symbol* $\neg$, *and the two constant symbols* $0, 1$.
Every Boolean formula is naturally viewed as a tree, except for variables $x \in \mathcal{X}$ *which are encoded as trees* $x(0, 1)$ *over* $\Sigma_{\mathcal{X}}$. *For instance, the formula* $(x \wedge y) \vee \neg x$ *is encoded as the tree* $\vee(\wedge(x(0, 1), y(0, 1)), \neg(x(0, 1)))$.

[a]Note that if there is a state q such that $q \neq_A q$, then any run in which q occurs is unsuccessful, as it would require that the subtree evaluated to q is different from itself. The tree automata with global constraints of [3] extend TAGEDs in the following way: a constraint is a pair of nodes $(u, v) \in {N_t}^2$ such that, as for TAGEDs, $r(u) =_A r(v)$ or $r(u) \neq_A r(v)$, but with the additional requirement that $u \neq v$. This leads to extra expressive power, as for instance an unbounded set of subtrees can be tested to be pairwise different, which is not possible with TAGEDs.

Let $Q = \{q_x \mid x \in \mathcal{X}\} \cup \{q_0, q_1, p_0, p_1\}$*, and* $F = q_1$*. The idea is to choose non-deterministically to evaluate the leaf* 0 *or* 1 *below* x *to* q_x*, but not both, for all* $x \in \mathcal{X}$*. It means that we assign* 0 *or* 1 *to a particular occurrence of* x*. Then, by imposing that every leaf evaluated to* q_x *are equal (by the constraints* $q_x =_{A_{sat}} q_x$*), for all* $x \in \mathcal{X}$*, we can ensure that we have chosen to same Boolean value for all occurrences of* x*. This can be done with the following rules:*

$$b \to p_b \qquad \neg(q_b) \to q_{\neg b} \qquad \oplus(q_{b_1}, q_{b_2}) \to q_{b_1 \oplus b_2} \qquad \forall b, b_1, b_2 \in \{0,1\}, \forall \oplus \in \{\wedge, \vee\}$$
$$b \to q_x \qquad x(p_0, q_x) \to q_1 \qquad x(q_x, p_1) \to q_0 \qquad \forall b \in \{0,1\}, \forall x \in \mathcal{X}$$

Proposition 4 (Uniform Membership [11, 16]) *The following problem is NP-Complete (even for RTAs): given a TAGED* A *and a tree* t*, decide if* $t \in \mathcal{L}(A)$*.*

One can see a TAGED as a computational machine which runs on trees in a bottom-up fashion. A TAGED is therefore said to be *deterministic* if there is at most one possible computation (not necessarily successful) per trees. This notion can be defined as usual with a simple syntactic restriction which says that all rules have different left-hand sides. Note that for all deterministic TAGED A, it is possible to compute a non-deterministic TAGED accepting the complement of $\mathcal{L}(A)$: we have to check if the tree evaluates in a non-accepting state or in an accepting state but in this case we non-deterministically guess a position where a constraint is not satisfied. However:

Proposition 5 ([13, 11]) *TAGEDs are not determinizable.*

Deterministic TAGEDs are still strictly more expressive than tree automata, since $\{f(g(t), g(t)) \mid t \in T_{\{f,a\}}\}$ is definable by a deterministic TAGED$^+$ but not by a tree automaton. It suffices to evaluate every tree of $T_{\{f,a\}}$ in a state q, and to add the rules $g(q) \to q_=$ and $f(q_=, q_=) \to q_f$, for some state q and final state q_f, where $q_= =_A q_=$. Proposition 5 is not surprising, since:

Proposition 6 ([13, 11]) *TAGED-definable languages are closed under union and intersection, but not by complement.*

Proof. Closure under union and intersection have been proven in [13, 11]. Although the definition of TAGEDs is slightly more general than in [13, 11], the proofs of closure remain true. In particular, closure by union is obtained by taking the disjoint union of the two TAGEDs, and closure by intersection is obtained by taking the product automaton of the two TAGEDs. The equality relation is therefore defined on pair of states[b] where we set $(p_1, q_1) =_{A_1 \times A_2} (p_2, q_2)$ if $p_1 =_{A_1} p_2$ or $q_1 =_{A_2} q_2$, and $(p_1, q_1) \neq_{A_1 \times A_2} (p_2, q_2)$ if $p_1 \neq_{A_1} p_2$ or $q_1 \neq_{A_2} q_2$.

Here we focus on complementation. We exhibit a tree language whose complement is easily definable by a TAGED, but which is not TAGED-definable. The idea

[b]Note however that if the two automata are RTAs, the product of them may not be an RTA, but it can be transformed into an RTA possibly exponentially larger (Proposition 9)

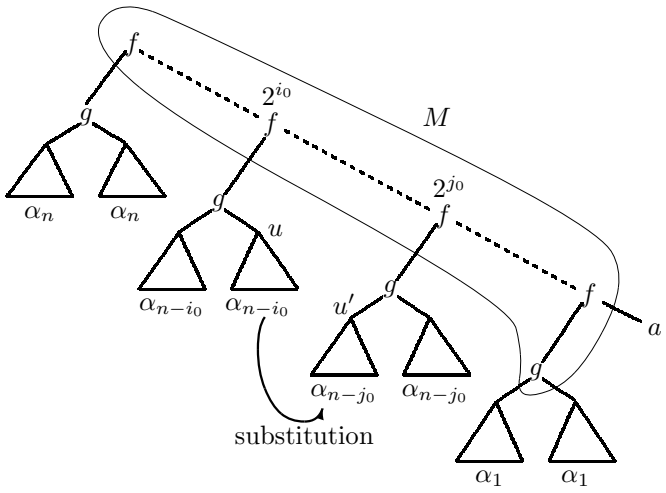

Fig. 1: Tree t in the proof of Proposition 6

is to take a language where at each level, the two subtrees are equal. To check membership to its complement, it suffices to guess a position where the two subtrees are different, which can be done by a TAGED.

Let $\Sigma = \{f, g, a\}$ where f, g are binary and a is a constant, and $h \notin \Sigma$ be a binary symbol. Let $T_0 = \{a\}$, and for $n > 0$, $T_n = \{f(g(t,t),t') \mid t \in T_{\{h,a\}},\ t' \in T_{n-1}\}$. Let $L = \bigcup_{n\in\mathbb{N}} T_n$. The complement of L is easily definable by a TAGED. Indeed, let L' be the regular language which satisfies $t \in L'$ iff there exist $t' \in L'$ and $t_1, t_2 \in T_{\{h,a\}}$ such that $t = f(g(t_1,t_2),t')$. It is easy to construct a tree automaton A' which defines L'. The TAGED that defines the complement of L has to check that the input tree does not belong to L', or if it belongs to L', then there is a node labeled by g such that its two children t_1 and t_2 are such that $t_1 \neq t_2$. This can be done by non-deterministically choosing two children of a node g and checking their difference. Therefore one needs only one disequality constraint to define the complement of L.

Suppose now that L is definable by a TAGED $A = (\Sigma \cup \{h\}, Q, F, \Delta, =_A, \neq_A)$. Let $n \geq |Q| + 1$, and let $\alpha_1, \ldots, \alpha_n \in T_{\{h,a\}}$ such that $\forall i < j, \|\alpha_i\| < \|\alpha_j\|$. Now, let $t_0 = a$, and for $i > 0$, $t_i = f(g(\alpha_i, \alpha_i), t_{i-1})$ (see Fig 1). Let $t = t_n$. It is clear that $t_n \in T_n$. Hence there is a successful run r of A on t_n. For all words $\pi \in \{1,2\}^*$ we denote by $node_t(\pi)$ the node of t that can be reached from the root by following the directions given by π. E.g. $node_t(12)$ is the second child of the first child of the root. Since $n \geq |Q| + 1$, there are $b, b' \in \{1,2\}$, $i_0, j_0 \in \{0, \ldots, n-1\}$, $i_0 < j_0$, two nodes $u, u' \in N_t$, and a state $q \in Q$ such that: (i) $u = node_t(2^{i_0}1b)$ and $u' = node_t(2^{j_0}1b')$, (ii) $r(u) = r(u') = q$. Let $t' = t[t|_u]_{u'}$, i.e. the tree t where the subtree at node u' has been substituted by the subtree at node u (see Fig 1). We do the same corresponding substitution in r, which results in a run denoted r'. Note that $t' \notin L$ since $\|\alpha_i\| \neq \|\alpha_j\|$ by definition of t, for all $i \neq j$. We now prove that r'

satisfies the constraints, which will contradict $t|_u \neq t|_{u'}$. Let $M = \{node_{t'}(2^k) \mid k = 0, \ldots, j_0\} \cup \{node_{t'}(2^{j_0}1)\}$ (see Fig 1). Intuitively, M is the set of nodes u_1 such that a constraint with u_1 is potentially unsatisfied, i.e. there is a node u_2 such that (u_1, u_2) is an unsatisfied constraint. Let $(v, w) \in \mathrm{cst}_=(t', r') \cup \mathrm{cst}_{\neq}(t', r')$ such that $v \neq w$ (the case $v = w$ is trivial). We consider three cases:

Case 1: if $v, w \notin M$, then it may be the case that v or w is a new node that comes from the duplication of the subtree $t|_u$. In all cases, there are $v', w' \in N_t$ such that $t|_{v'} = t'|_v$ and $t|_{w'} = t'|_w$, $r'(v) = r(v')$, $r'(w) = r(w')$, and the constraint (v', w') is satisfied in t and r. Therefore the constraint (v, w) is also satisfied in t' and r';

Case 2: $v \in M$ and $v = node_{t'}(2^k)$, for some $k \leq j_0$.

Case 2.1: Suppose $r'(v) \neq_A r'(w)$. We prove $t'|_v \neq t'|_w$. Indeed, the root of $t'|_v$ is necessarily labeled by f. If the root of $t'|_w$ is labeled by f, then either $t'|_v$ is a strict subtree of $t'|_w$ or $t'|_w$ is a strict subtree of $t'|_v$. Otherwise the root of $t'|_w$ is labeled by g, h or a, so we obviously have $t'|_v \neq t'|_w$.

Case 2.2: Suppose $r'(v) =_A r'(w)$. We prove a contradiction. If $w \in M$, by definition of the pumping $M \subseteq N_t$, so that $t|_v$ and $t|_w$ are well-defined. Then we have $\|t'|_v\| = \|t|_v\| - \|\alpha_{n-j_0}\| + \|\alpha_{n-i_0}\|$ and $\|t'|_w\| = \|t|_w\| - \|\alpha_{n-j_0}\| + \|\alpha_{n-i_0}\|$, so that $\|t'|_v\| - \|t|_v\| = \|t'|_w\| - \|t|_w\|$. Since the constraints were satisfied in t and r, we get $\|t|_v\| = \|t|_w\|$, so that $\|t'|_v\| = \|t'|_w\|$, which is impossible since $v \neq w$ and v and w are comparable. If $w \notin M$, then $t'|_w$ does not contain any node labeled by f. Moreover, by definition of the pumping, there exists some node $w' \in N_t$ such that $t'|_w = t|_{w'}$ and $r(w') = r'(w)$. Since $v \in M$, $t|_v$ exists and contains a node labeled by f. Since r satisfies the constraints, $t|_w = t|_v$, which contradicts the fact that $t|_w$ does not contain any node labeled by f.

Case 3: $v \in M$ and $v = node_{t'}(2^{j_0}1)$. Necessarily, $t'(v) = g$. There are two cases:

Case 3.1: $r'(v) =_A r'(w)$. We can prove a contradiction. Indeed:

If $w = node_{t'}(2^{j_0}1\pi)$, for some $\pi \in \{1, 2\}^+$, then by definition of the pumping and the fact that the constraints were satisfied in t and r, $t|_{node_t(2^{i_0}1\pi)} = t'|_{node_{t'}(2^{j_0}1\pi)} = t|_{node_t(2^{j_0}1)}$. This contradicts the fact that $t|_{node_t(2^{i_0}1\pi)}$ does not contain any node labeled by g.

If $w = node_{t'}(2^k)$, for some $k \in \{0, \ldots, n\}$, then we have $t|_w = t|_v$, which is also impossible for similar reasons.

If $w = node_{t'}(2^k1)$, for some $k \in \{0, \ldots, n-1\}$, then $w = node_t(2^k1)$, $k \neq j_0$, and necessarily we have $t|_w = t|_v$. It is impossible since it implies that $t|_w = g(\alpha_{n-k}, \alpha_{n-k})$ and $t|_v = g(\alpha_{n-j_0}, \alpha_{n-j_0})$, which contradicts $|\alpha_{n-k}| \neq |\alpha_{n-j_0}|$.

If $w = node_{t'}(2^j1\pi)$, for some $j \neq j_0$ and $\pi \in \{1, 2\}^+$, then by definition of the pumping, we get $w = node_t(2^j1\pi)$, and $\mathsf{lab}^r(v) =_A \mathsf{lab}^r(w)$, and $t|_v = t|_w$, which contradicts that their respective roots are labeled by different labels.

Case 3.2: $r(v) \neq_A r(w)$. Then the constraint is satisfied, i.e. $t'|_v \neq t'|_w$. Indeed, suppose that $t'|_v = t'|_w$. Since the root of $t'|_v$ is labeled by g, w is necessarily equal to $node_{t'}(2^k1) = node_t(2^k1)$, for some $k \neq j_0$. Hence $t|_w$ is of the form $g(\alpha_{n-i_0}, \alpha_{n-j_0}) = t'|_v$, which contradicts $t \in L$, since $\|\alpha_{n-i_0}\| \neq \|\alpha_{n-j_0}\|$. □

	TA	RTA	TAGED$^+$	TAGED$^-$	vbTAGED	TAGED
$\bigcup$	yes	yes	yes	yes	yes	yes
$\bigcap$	yes	yes	yes	yes	yes	yes
$\neg$	yes	no	no	no	no	no
membership	P	NP-C	NP-C	NP-C	NP-C	NP-C
emptiness	P	P	EXP-C	NEXP	2NEXP	dec[3]
universality	EXP-C	undec	undec	undec	undec	undec
determinizable	yes	no	no	no	no	no

Fig. 2: Summary of closure properties and decision problems

These (non)closure properties also hold for the class of languages defined by TAGED$^+$s or TAGED$^-$s (rigid or not). The complement of the tree language L used as a counter-example in the proof of Proposition 6 is definable by a TAGED$^-$. This means in particular that the class of languages definable by TAGED$^-$s is not closed by complement. One could also use a symmetric counter-example language L' to prove that the class of languages definable by TAGED$^+$s is not closed by complement. Instead of requiring an equality between the children of nodes labeled by g, it would suffice to require a disequality.

The universality problem is as follows: given a TAGED A over a ranked alphabet Σ, does A accept all trees over Σ?

Proposition 7 ([11, 16]) *Testing universality of TAGED (resp. RTA, TAGED$^+$, vbTAGED) is undecidable.*

The proof given in [11] relies on the use of a bounded number of negative and positive tests on unary trees. It implies undecidability for vbTAGED (defined in Section 6). In [16], it is shown that positive constraints can be used to test inequalities in unary trees (one has to guess the highest positions where the two subtrees are equal and test if their parent labels are different). In particular, this proves undecidability for RTA, and therefore for TAGED$^+$. The reduction of [11] is based on the Post correspondence problem (PCP). By using a variant of PCP called the 2-*marked Post correspondence problem* [15], the equality constraints in the reduction of [11] become useless, so that we get undecidability for TAGED$^-$ as well.

A table of the known results for various classes of TAGEDs is given in Fig. 2. In this table, TA stands for tree automata and vbTAGED for a class of TAGEDs that mixes equalities and disequalities, introduced in Section 6. Other acronyms have been defined in Section 3. The given complexity bounds are time complexities. Emptiness for TAGED$^+$s, TAGED$^-$s, and TAGEDs is studied in following sections.

4. TAGEDs and Rigid TAGEDs

We prove that TAGEDs and rigid TAGEDs have the same expressiveness.

Node Equivalence Given a tree $t \in \mathcal{L}(A)$ and a run r of A on t which satisfies the equality constraints, we define an equivalence relation $\sim_{t,r}$ on N_t as follows[c]. The relation $\sim_{t,r}$ is the transitive and reflexive closure of the relation $\leftrightarrow_{t,r}$ defined as follows: for all $u, v \in N_t$, $u \leftrightarrow_{t,r} v$ if there exist $u', v' \in N_t$ such that $u' \lhd^* u$, $v' \lhd^* v$, $\text{path}_t(u', u) \equiv \text{path}_t(v', v)$ and $r(u') =_A r(v')$. For instance, Fig 3 shows a tree where the two subtrees have been evaluated to some state q such that $q =_A q$, and the corresponding equivalence relation (reflexivity is not depicted in the figure). We denote by $[u]_{t,r}$ the equivalence class of any node u by $\sim_{t,r}$.

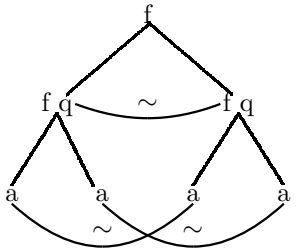

Fig. 3: Equivalence relation where $q =_A q$

It is easy to prove the following proposition:

Proposition 8. *For all $u, v \in N_t$, if $u \sim_{t,r} v$, then $t|_u = t|_v$. For all $u', v' \in N_t$ such that $u \lhd^* u'$ and $v \lhd^* v'$, if $u \sim_{t,r} v$ and $path_t(u, u') \equiv path_t(v, v')$, then $u' \sim_{t,r} v'$.*

Proposition 9. *For every TAGED (resp. TAGED$^+$) A, one can construct an equivalent reflexive TAGED (resp. TAGED$^+$) A' in exponential time, whose size is possibly exponential in the size of A. Moreover, A' is obtained as the union of exponentially many reflexive TAGED A_i of polynomial size (in $\|A\|$).*

Proof. Let $A = (\Sigma, Q, F, \Delta, =_A, \neq_A)$. For all $P \subseteq Q$, we construct a TAGED $A_P = (\Sigma, Q_P, F_P, \Delta_P, =_P, \neq_P)$. We let $Q_P = P$, Δ_P is the restriction of Δ to the rules in which only states of P occur, $F_P = F \cap P$, $\neq_P |_{P \times P}$ (the relation $\neq_P$ restricted to pairs of $P \times P$, and $=_P$ is defined by $=_A |_{P \times P} \cup \{(p, p) |\ p \in \text{dom}(=_A |_{P \times P})\}$. We let A' such that $\mathcal{L}(A') = \bigcup_{P \subseteq Q} \mathcal{L}(A_P)$ (it exists by effective closure by union).

$\mathcal{L}(A') \subseteq \mathcal{L}(A)$. Let $P \subseteq Q$, $t \in \mathcal{L}(A_P)$ and r a successful run of A_P on t. The run r is also a run of A on t (since A_P is just a restriction of A). Moreover, it satisfies the constraints. Indeed, let $u, v \in N_t$ such that $r(u) =_A r(v)$. By definition of $=_P$, we also have $r(u) =_P r(v)$ and since r is successful, $t|_u = t|_v$. It is similar for disequality constraints.

$\mathcal{L}(A) \subseteq \mathcal{L}(A')$. Let $t \in \mathcal{L}(A)$ and r a successful run of A on t. Let $P = \{r(u) \mid u \in N_t\}$ the set of states occuring in r. We now prove that $t \in \mathcal{L}(A_P)$, by proving that r is also a successful run of A_P on t. It suffices to prove that t satisfies the constraints. Let $u, v \in N_t$ such that $r(u) =_P r(v)$. If $r(u) =_A r(v)$, then $t|_u = t|_v$ since r is successful in A. Otherwise by definition of $=_P$, we have necessarily $r(u) = r(v) = p$, for some $p \in P$, and there exists $q \in P$ such that $p =_A q$. Since $q \in P$, q occurs in r at some node $w \in N_t$. Since r satisfies the equality constraints in A, we have $t|_w = t|_u = t|_v$. It is also clear that r satisfies the disequality constraints induced by $\neq_P$ since $\neq_P$ is just a restriction of $\neq_A$. □

[c]when it is clear from the context, we omit the subscript t, r and write $\sim$

Theorem 10. *For all reflexive TAGED A (resp. reflexive TAGED$^+$ A), one can construct an equivalent rigid TAGED A' (resp. RTA) in exptime, whose size is possibly exponential in the size of A.*

Proof. Intuitively, we can view an accepting run r of A on a tree t as a DAG structure. Let $U \subseteq N_t$ such that all subtrees $t|_u$, $u \in U$, have been successfully tested to be equal by A in the run r (i.e. $\forall u, v \in U$, $r(u) =_A r(v)$). Let $t_0 = t|_u$, for some $u \in U$. We replace all nodes of U by a single node u_0 which enroots t_0. The parent of any node of U points to u_0. We maximally iterate this construction to get the DAG. Note that this DAG is not maximal sharing[d], since only subtrees which have been successfully tested to be equal are shared. We construct A' s.t. it simulates a run on this DAG, obtained by overlapping the runs on every equal subtrees for which a test has been done. We now formally define the construction of A' and then prove its correctness.

Automata construction We define $Q' = 2^Q \times \mathcal{C}$ where $\mathcal{C}$ is a set of choices depending on A. Intuitively, when a choice has been made, it enforces the subtrees successfully tested to be equal to run in the same state. This is done by grouping the states which are used at equivalent positions in the tree. Formally, a *choice* is a (partial) function $c : \mathrm{dom}(=_A) \to 2^Q$ such that for all $q, q' \in \mathrm{dom}(c)$, $q =_A q'$ implies $c(q) = c(q')$. Note that for all $q \in \mathrm{dom}(c)$, $c(q)$ may contain states from $Q \backslash \mathrm{dom}(=_A)$. For every choice $c \in \mathcal{C}$, we define a TAGED $A_c = (\Sigma, Q_c, F_c, \Delta_c, =_c, \neq_c)$. We let $Q_c = \{P \subseteq Q \mid \forall q.(q \in P \cap \mathrm{dom}(=_A) \implies P = c(q))\}$ (it imposes that P must respect the choice c). We let Δ_c be the set of rules defined as follows: (i) for all states $P, P', P'' \in Q_c$, $f(P, P') \to P'' \in \Delta_c$ iff for all $p'' \in P''$, there are $p \in P$ and $p' \in P'$ such that $f(p, p') \to p'' \in \Delta$; (ii) for all $P \in Q_c$, $a \to P \in \Delta_c$ iff for all $p \in P$, we have $a \to p \in \Delta$. The set of final states F_c is defined by $F_c = \{P \in Q_c \mid P \cap F \neq \emptyset\}$. Finally, we define $=_c, \neq_c$ by:

$$P =_c P' \text{ if } \exists p \in P,\ \exists p' \in P',\ p =_A p' \quad (\text{ hence } P = P')$$
$$P \neq_c P' \text{ if } \exists p \in P, \exists p' \in P', p \neq_A p'$$

We let A' be the TAGED accepting $\bigcup_{c \in \mathcal{C}} \mathcal{L}(A_c)$ (we can construct it thanks to Proposition 6, and its size is exponential in the size of A, and rigidness is preserved by union of automata).

Correctness We now prove that $\mathcal{L}(A') = \mathcal{L}(A)$. Let $q \in Q$. We let $\mathcal{L}(q, A)$ be the set of trees t such that there exists a q-run[e] of A on t which satisfies the constraints. $\mathcal{L}(q, A')$ is defined similarly.

· $\mathcal{L}(A') \subseteq \mathcal{L}(A)$. Let $c \in \mathcal{C}$, $P \in Q_c$, and $t \in T_\Sigma$ such that $t \in \mathcal{L}(P, A_c)$. By definition of Δ_c, we can easily prove by induction on t that if there is a P-run r_c of A_c on t, then for all $p \in P$, there exists a p-run r of A on t. The run r is constructed

[d]There might be two isomorphic subgraphs occurring at different positions.
[e]A q-run is a run whose root is labeled by q

inductively in a top-down fashion. One first chooses a final state in the root of r_c. Then one chooses two states in the respective two subtrees of the root according to the existence of a rule of Δ, and so on until reaching the leaves of r_c. Moreover, if r_c satisfies the constraints, then r satisfies the constraints too. Indeed, let $u, v \in N_t$ such that $r(u) =_A r(v)$. By construction of r, we have $r(u) \in r_c(u)$ and $r(v) \in r_c(v)$. By definition of $=_c$, $r_c(u) =_c r_c(v)$ and so $t|_u = t|_v$. It is similar for inequalities.

· $\mathcal{L}(A) \subseteq \mathcal{L}(A')$. Let $t \in \mathcal{L}(A)$ and r be a successful run of A on t. We let $\text{states}(u) = \{q \mid \exists v \sim u, q = r(v)\}$. We let $\mathfrak{c}$ be defined as follows: for all $p \in \text{dom}(=_A)$, $u \in N_t$, we set $\mathfrak{c}(p) = \text{states}(u)$ if $p \in \text{states}(u)$. We have to prove that $\mathfrak{c}$ is well-defined. Let $u, v \in N_t$ and $p \in \text{dom}(=_A)$ such that $p \in \text{states}(u) \cap \text{states}(v)$. We prove that $\text{states}(u) = \text{states}(v)$. There exist $u', v' \in N_t$ such that $u' \sim u$, $v' \sim v$ and $r(u') = r(v') = p$. Since $=_A$ is reflexive, we have $p =_A p$, and by definition of $\sim$, we get $u' \sim v'$, and $u \sim v$. Therefore $\text{states}(u) = \text{states}(v)$. Let $p, q \in \text{dom}(c)$ such that $p =_A q$. By definition of $\mathfrak{c}$, there exists u, v such that $\text{states}(u) = \mathfrak{c}(p)$ and $\text{states}(v) = \mathfrak{c}(q)$, $r(u) = p$ and $r(v) = q$. Therefore $u \sim v$, and $\text{states}(u) = \text{states}(v)$, i.e. $\mathfrak{c}(q) = \mathfrak{c}(p)$. Now, for all $u \in N_t$, we let $r_c(u) = \text{states}(u)$. The tree r_c fulfills all the conditions to be a successful run of A_c on t, indeed:

(*i*) *for all* $u \in N_t$, $r_c(u) \in Q_c$. This holds by definition of $\text{states}(u)$ and Q_c;

(*ii*) r_c *is a run of* A_c *on* t. Let $u, u_1, u_2 \in N_t$ such that u_1 and u_2 are the sons of u. We show that the transition $f(\text{states}(u_1), \text{states}(u_2)) \to \text{states}(u)$ is in Δ_c. Let $p \in \text{states}(u)$. There is some node $v \in N_t$ such that $u \sim v$ and $r(v) = p$. Let v_1, v_2 be the two sons of v respectively (they exist since $t|_u = t|_v$). Let $p_1 = r(v_1)$ and $p_2 = r(v_2)$. Hence there is a rule of the form $f(p_1, p_2) \to p$ in Δ. By definition of $\sim$, we have $u_1 \sim v_1$ and $u_2 \sim v_2$, hence $p_1 \in \text{states}(u_1)$ and $p_2 \in \text{states}(u_2)$, which concludes the proof (this goes similarly for leaf nodes).

(*iii*) r_c *satisfies the equality constraints.* Let $u_1, u_2 \in N_t$ and let $P_1, P_2 \in Q_c$ such that $r_c(u_1) = P_1$ and $r_c(u_2) = P_2$. Suppose that $P_1 =_c P_2$. It means that there are two states $p_1 \in P_1$ and $p_2 \in P_2$ such that $p_1 =_A p_2$. Hence, $\mathfrak{c}(p_1) = \mathfrak{c}(p_2) = P_1 = P_2 = \text{states}(u_1) = \text{states}(u_2)$. Hence, there are $u_1', u_2' \in N_t$ (not necessarily different from u_1 and u_2) such that $u_1' \sim u_1$ and $u_2' \sim u_2$, and $r(u_1') = p_1$, $r(u_2') = p_2$. By definition of $\sim$, we also get $u_1' \sim u_2'$, since $p_1 =_A p_2$. Finally, as $\sim$ is transitive, we get $u_1 \sim u_2$, and $t|_{u_1} = t|_{u_2}$.

(*iv*) r_c *satisfies the disequality constraints.* It is similar to the previous case. Let $u_1, u_2 \in N_t$. If $r_c(u_1) \neq_c r_c(u_2)$, it means that there are two nodes $u_1' \sim u_1$ and $u_2' \sim u_2$ such that $r(u_1') \neq_A r(u_2')$. Since $t|_{u_1} = t|_{u_1'}$ and $t|_{u_2} = t|_{u_2'}$, and $t|_{u_1'} \neq t|_{u_2'}$, we get $t|_{u_1} \neq t|_{u_2}$. □

As a corollary of Proposition 9 and Theorem 10:

Corollary 11. *For all TAGED (resp. TAGED$^+$), one can construct (in exponential time) an equivalent rigid TAGED (resp. RTA) of possibly exponential size.*

When a TAGED is rigid, we can define a normal form for the runs that satisfy the equality constraints: the subruns below equivalent nodes are equal. Formally:

Lemma 12. *Let A be a rigid TAGED. Let $t \in T_\Sigma$. If there is a run r of A on t which satisfies the equality constraints, then there is a run r' of A on t such that:*

(1) r' satisfies the equality constraints;
(2) if r satisfies the disequality constraints, then r' satisfies the disequality constraints;
(3) for all $u, v \in N_r$, if $u \sim_{t,r'} v$, then $r'|_u = r'|_v$;
(4) $r(\epsilon_r) = r'(\epsilon_{r'})$.

Proof. In the first step, we show that we can equivalently weaken the third condition: for all $u, v \in N_r$, if $r'(u) =_A r'(v)$, then $r'|_u = r'|_v$. Indeed, by definition of $\leftrightarrow_{t,r}$, for all nodes u, v such that $u \leftrightarrow_{t,r} v$, there are u' and v' above u and v respectively such that $\mathrm{path}_t(u', u) \equiv \mathrm{path}_t(v', v)$, and $r(u') =_A r(v')$. If this weaker condition holds, we get $r|_{u'} = r|_{v'}$, from which we deduce $r|_u = r|_v$. Since $\sim_{t,r}$ is the reflexive and transitive closure of $\leftrightarrow_{t,r}$, by transitivity, we also get $r|_u = r|_v$ for all nodes u, v such that $u \sim_{t,r} v$.

Now, the construction of r' is done via the following rewriting algorithm:

(1) $q_1, \ldots, q_n \leftarrow \mathrm{dom}(=_A)$ (the order is chosen arbitrarily)
(2) $r_0 \leftarrow r$
(3) **for** $i = 1$ to n **do**
(4) $\quad U_i \leftarrow \{u \in N_{r_{i-1}} \mid r_{i-1}(u) = q_i\}$
(5) $\quad u_i \leftarrow$ some node of U_i
(6) $\quad r_i \leftarrow$ for all $v \in U_i$, replace in r_{i-1} the subrun at node v by $r_{i-1}|_{u_i}$
(7) **end for**
(8) $r' \leftarrow r_n$
(9) **return** r'

As we next show, every run r_i satisfies the equality constraints, so that the substitution is well-defined, since all nodes of U_i are disjoint. We prove the following invariant (called $\mathcal{I}$): for all $i \in \{0, \ldots, n\}$, r_i is a run of A on t which satisfies the equality constraints and such that for all j such that $1 \leq j \leq i$, and all $u, v \in N_{r_i}$, if $r_i(u) = r_i(v) = q_j$, then $r_i|_u = r_i|_v$.

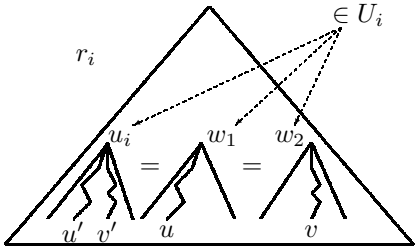

Fig. 4:

It is clearly true at rank 0 since $r_0 = r$. Let $i > 0$ and suppose that it holds at rank $i-1$. Since r_{i-1} satisfies the equality constraints, for all $v \in U_i$, $t|_{u_i} = t|_v$, hence $r_{i-1}|_{u_i}$ is also a run of A on $t|_v$, so that r_i is still a run of A on t. Since r_{i-1} satisfies the equality constraints, it basically maps every state q of $\mathrm{dom}(=_A)$ to at most one tree t_q (which is a subtree of t). The substitution preserves this property (no new mappings are created). Hence, since $=_A \subseteq id_Q$,

the equality constraints are still satisfied in r_i. Finally, let $u, v \in N_{r_i}$ such that $r_i(u) = r_i(v) = q_j$, for some $j \in \{1, \ldots, i\}$. If $i = j$, then by definition of r_i, we have $r_i|_u = r_i|_v$. If $j < i$, then we consider several cases (it is not exhaustive as other cases are symmetric):

• u is above (at least) one node of U_i. Hence v cannot be below U_i (otherwise $\|t|_u\| > \|t|_v\|$, since the equality constraints are satisfied). By induction hypothesis, $r_{i-1}|_u = r_{i-1}|_v$. Hence their respective positions labeled by q_i are isomorphic, so that the substitution is made at isomorphic positions, and we get $r_i|_u = r_i|_v$;

• u is below some $w_1 \in U_i$ and v is below some element $w_2 \in U_i$.

Since $r_i|_{w_1} = r_i|_{w_2} = r_i|_{u_i}$, we can define u' the nodes below u_i such that $\mathrm{path}_t(w_1, u) \equiv \mathrm{path}_t(u_i, u')$. Similarly, we can define v' such that $\mathrm{path}_t(w_2, v) \equiv \mathrm{path}_t(u_i, v')$. This situation is depicted in Fig. 4. Hence $r_i|_{u'} = r_i|_u$ and $r_i|_{v'} = r_i|_v$. Therefore u' and v' are labeled by q_j in r_i and r_{i-1}. Hence by induction hypothesis, $r_{i-1}|_{u'} = r_{i-1}|_{v'}$, and by definition of the substitution, $r_{i-1}|_{u'} = r_i|_{u'}$ and $r_{i-1}|_{v'} = r_i|_{v'}$. Consequently $r_i|_{u'} = r_i|_{v'}$, and we get $r_i|_u = r_i|_v$;

• u is below some $w \in U_i$ but v is incomparable to any node of U_i. In this case the argument is similar to the latter case. We can define u' the node below u_i such that $\mathrm{path}_t(u_i, u') \equiv \mathrm{path}_t(w, u)$, and get $r_i|_u = r_i|_v$ since by induction hypothesis, $r_{i-1}|_{u'} = r_{i-1}|_v$;

• u and v are both incomparable to any node of U_i. In this case the subruns rooted at u and v remain unchanged by the substitution, *i.e.* $r_{i-1}|_u = r_i|_u$ and $r_{i-1}|_v = r_i|_v$. Hence by induction hypothesis, we get $r_i|_u = r_i|_v$. □

5. Positive and Negative TAGEDs

In this section we recall results on TAGED$^+$s and prove emptiness of TAGED$^-$s, by reducing the problem to testing satisfiability of set constraints. We prove in [13] that emptiness of TAGED$^+$ is EXPTIME-complete. The upper-bound is obtained by first going to an RTA and then by applying the classical emptiness algorithm for tree automata. The lower-bound is obtained by reduction from the intersection of n tree automata.

Theorem 13. *Deciding emptiness of a TAGED$^+$ A is EXPTIME-complete, and in linear time if A is rigid. Moreover, if $\mathcal{L}(A) \neq \varnothing$, then a tree $t \in \mathcal{L}(A)$ is computable in EXPTIME, and in linear time if A is rigid.*

Again by reduction to RTAs and by the result of [16], also proved in [11]:

Theorem 14. *Let A be a TAGED$^+$. It is decidable whether $\mathcal{L}(A)$ is infinite or not, in $O(\|A\|.|Q|^2)$ if A is rigid, and in EXPTIME otherwise.*

We now prove decidability of emptiness for TAGED$^-$s, by reduction to positive and negative set constraints (PNSC for short). Set expressions are built over set variables, function symbols, and Boolean operators. Set constraints are either positive, $e_1 \subseteq e_2$, or negative, $e_1 \not\subseteq e_2$, where e_1, e_2 are set expressions. Set expressions are

interpreted in the Herbrand structure (where set variables are therefore interpreted by sets of terms) while set constraints are interpreted by Booleans 0,1. Testing the existence of a solution of a system of set constraints has been proved to be decidable in several papers [7, 1, 22, 14]. In particular, it is known to be NEXPTIME-complete. We do not formally define set constraints and refer the reader to [7, 1, 22, 14].
Consider for instance the constraint $f(X, X) \subseteq X$. It has a unique solution which is the empty set. Consider now $X \subseteq f(X, X) \cup a$, where a is a constant symbol. Every set of terms over $\{f, a\}$ closed by the subterm relation is a solution of this equation. More generally, we can encode the emptiness problem of tree automata as a system of set constraints. Let $A = (\Sigma, Q, F, \Delta)$ be a tree automaton. Wlog, we assume that every state $q \in Q$ occurs in the rhs of some rule. We associate with A the system S_A defined by:

$$(S_A) \quad \begin{cases} X_q \subseteq \bigcup_{f(q_1,q_2)\to q\in\Delta} f(X_{q_1}, X_{q_2}) \cup \bigcup_{a\to q\in\Delta} a & \text{for all } q \in Q \\ \bigcup_{q\in F} X_q \not\subseteq \varnothing \end{cases}$$

The following result is well-known:

Proposition 15 ([23]) *$\mathcal{L}(A)$ is non-empty iff S_A has a solution.*

Proof. We sketch correctness of the system S_A. Suppose that S_A has a solution given by a set of trees T_q, for all $q \in Q$. Let $q \in Q$ such that there is $t \in T_q$, we construct a run of A on t inductively. If $t = a \in \Sigma$, then we take the run reduced to the leaf q. If $t = f(t_1, t_2)$, for some $f \in \Sigma$ and $t_1, t_2 \in T_\Sigma$, then by definition of S_A, there is a rule $f(q_1, q_2) \to q$ such that $t_1 \in T_{q_1}$ and $t_2 \in T_{q_2}$. By induction hypothesis, there are runs r_1 and r_2 on t_1 and t_2 respectively, such that $r_1(\epsilon_{r_1}) = q_1$ and $r_2(\epsilon_{r_2}) = q_2$. Thus $q(r_1, r_2)$ is a run of A on t.

Since there is $q \in F$ such that $T_q \neq \varnothing$, for all $t \in T_q$, we can construct a successful run r of A on t, so that $t \in \mathcal{L}(A)$. This proves the first direction.

Conversely, if $\mathcal{L}(A) \neq \varnothing$, there is a tree $t \in \mathcal{L}(A)$ and a successful run r of A on t. For all $q \in Q$, we let $T_q = \{t|_u \mid u \in N_t, r(u) = q\}$. The set $\{T_q \mid q \in Q\}$ is a solution of S_A. □

Let $(A, \neq_A)$ be a TAGED$^-$, and consider the system S'_A consisting in S_A extended with the constraints $X_q \cap X_p = \varnothing$, for all $q, p \in Q$ such that $q \neq_A p$. It is easy to extend the proof of Proposition 15 to prove that $\mathcal{L}(A, \neq_A) \neq \varnothing$ iff S'_A has a solution. Since deciding existence of a solution of a system of PNSC is in NEXPTIME, we get:

Theorem 16. *Emptiness of TAGED$^-$s is decidable in NEXPTIME.*

6. Vertically Bounded TAGEDs

In this section, we define a subclass of TAGEDs, called *vertically bounded TAGEDs*, with both equality and disequality constraints and for which we can decide empti-

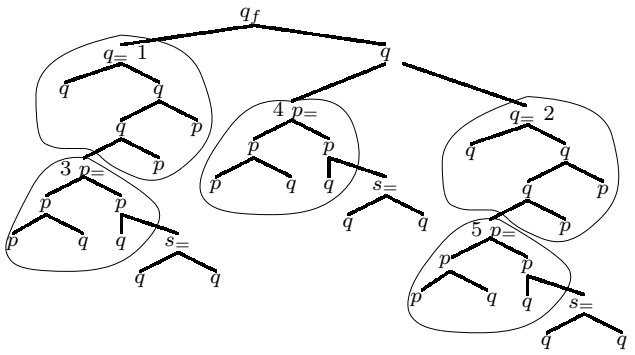

Fig. 5: Elementary contexts of a run r over $\{q, p, q_f, q_=, p_=, s_=\}$ where $q_= =_A q_=$, $p_= =_A p_=$ and $s_= =_A s_=$. Their root nodes are identified by natural numbers. The set $\mathcal{C}(r)$ is equal to $\{1, 2, 3, 4, 5\}$, the contexts rooted at 1 and 2, and 3,4,5 respectively are isomorphic. The subtrees rooted at $s_=$ are not elementary contexts since they do not contain a loop.

ness. In this class, only a bounded number of disequality tests is permitted along any root-to-leaf path.

Definition 17. *Let Σ be a ranked alphabet. A vertically bounded TAGED over Σ is a pair (A, k) where A is a TAGED over Σ, and $k \in \mathbb{N}$. A run r of (A, k) on a tree $t \in T_\Sigma$ is a run of A on t. It is successful if r is successful for A and the number of states from $dom(\neq_A)$ occurring along any root-to-leaf path is bounded by k:*

$$\textit{for all root-to-leaf path } u_1 \lhd^* \ldots \lhd^* u_n,\ |\{i \mid r(u_i) \in dom(\neq_A)\}| \leq k$$

The notion of the defined language $\mathcal{L}(A, k)$ is defined similarly as for TAGEDs. In the sequel, we are interested in the non-emptiness problem. We first weaken the acceptance condition by introducing the notion of *seed*. Informally, a seed is a pair (t, r) where r is a run on t which satisfies the equality constraints but not necessarily all the disequality constraints. However, we introduce a sufficient condition on t and r such that we can transform them – or *repair* them – into a pair (t', r') where r' is successful on t'.

Given two multi-ary contexts C_1, C_2, we say that C_1 is *included* in C_2 if C_1 occurs in C_2, *i.e.* if there are some unary context C'_0 and some contexts $C'_1, \ldots, C'_n$ such that n is the arity of C_1, and $C_2 = C'_0[C_1[C'_1, \ldots, C'_n]]$. Let $t \in \mathcal{L}(A)$ and r a successful run of A on t. A context C of r is *elementary* if it is a maximal context (w.r.t. context inclusion) included in r, such that (i) all its nodes (except the root) are labeled in $Q - (\mathrm{dom}(=_A) \cup \mathrm{dom}(\neq_A))$, (ii) the root is labeled in $\mathrm{dom}(=_A) \cup \mathrm{dom}(\neq_A)$, (iii) there is a loop in C, *i.e.* two descendant nodes of C are labeled by the same state in r. We denote by $\mathcal{C}(r)$ the set of nodes which enroot an elementary context. For all nodes $u \in \mathcal{C}(r)$, we denote by $\mathrm{cxt}_r(u)$ the elementary context over Q rooted at u in r, and by $\mathrm{cxt}_t(u)$ the context of t over Σ rooted at u and edge-isomorphic to $\mathrm{cxt}_r(u)$. Fig. 5 represents a run r on some tree t, and its elementary contexts.

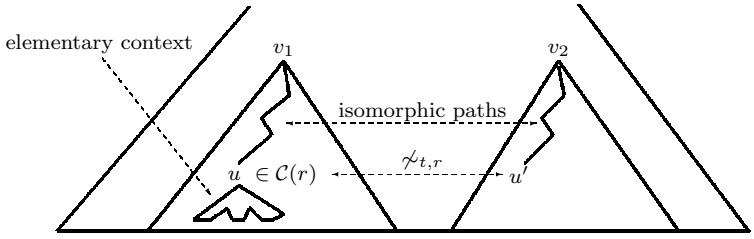

Fig. 6: A configuration where $\mathrm{Rep}(t, r, v_1, v_2, u)$ holds.

Definition 18. *Let $t \in T_\Sigma$ and $r \in T_Q$. The pair (t, r) is a seed if the following conditions are satisfied:*

(1) r is a run of A on t which satisfies the equality constraints, whose root is labeled by a final state, and such that in any $\lhd^*$-ordered chain, there are at most k nodes whose labels in r belong to $dom(\neq_A)$;
(2) for all $u, v \in N_r$, if $u \sim_{t,r} v$ then $r|_u = r|_v$;
(3) for all $(v_1, v_2) \in \mathrm{cst}^A_{\neq}(t, r)$, if $t|_{v_1} = t|_{v_2}$, then there is $u \in \mathcal{C}(r)$ such that $\mathrm{Rep}(t, r, v_1, v_2, u)$ holds (illustrated in Fig. 6), i.e.:
  (i) either $v_1 \lhd^* u$ or $v_2 \lhd^* u$;
  (ii) if $v_1 \lhd^* u$, then if $u' \in N_t$ is the node such that $v_2 \lhd^* u'$ and $path_t(v_1, u) \equiv path_t(v_2, u')$, then $u \not\sim_{t,r} u'$;
  (iii) if $v_2 \lhd^* u$, we define the condition symmetrically as (ii).

The degree of (t, r), denoted by $\deg(t, r)$, is the number of unsatisfied inequality constraints in (t, r),i.e. $\deg(t, r) = |\{(u, v) \in \mathrm{cst}^A_{\neq}(t, r) \mid t|_u = t|_v\}|$.

We now prove that a seed can indeed be repaired, and it can be done inductively. Before proving it formally, let us first introduce a general pumping idea, often used in the sequel. Let t and r be tree and a run respectively such that for all $u, v \in N_t$, if $u \sim_{t,r} v$, then $r|_u = r|_v$. Suppose also that the equality constraints are satisfied. The first idea of the pumping method is to transform subtrees (or subruns) below all equivalent nodes in parallel, at isomorphic positions. This is possible since the (sub)runs are the same below equivalent nodes. It ensures that equality constraints are still satisfied. We often refer to this as a *parallel* pumping technique.

Lemma 19 (Base Lemma) *If (t, r) is a seed of degree 0, then $t \in \mathcal{L}(A, k)$.*

Proof. By definition a seed always satisfies the equality constraints. □

Lemma 20 (Induction Lemma) *For all seeds (t, r) such that $\deg(t, r) > 0$, one can construct a seed (t', r') such that $\deg(t', r') < \deg(t, r)$.*

Proof. Informally, to obtain (t', r'), we repair some unsatisfied contraints in (t, r) while preserving equality constraints. This is done by choosing some $\sim_{t,r}$-equivalence class $\omega \subseteq \mathcal{C}(r)$, and by increasing in parallel the size of the elementary

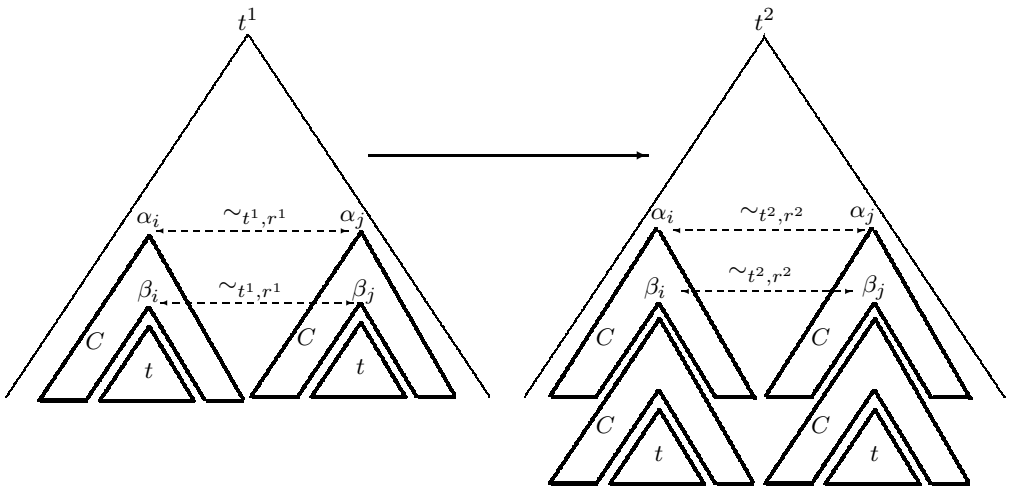

Fig. 7: Parallel pumping in elementary contexts

contexts in r (and t) rooted at nodes of ω, thanks to the loop they contain. This allows us to repair some unsatisfied inequality constraints that involve a node above some node of ω, while preserving equality constraints and the condition of being a seed. We can pump in a way that does not create new unsatisfied inequality constraints, making the degree strictly decreasing.

Construction of (t', r')**.** Since $\deg(t, r) > 0$, $\mathcal{C}(r) \neq \varnothing$, and there is a class of nodes $\omega \subseteq \mathcal{C}(r)$. Let $\omega = \{u_1, \ldots, u_n\}$. We first define formally how to increase the size of the elementary contexts rooted at u_is, while keeping the equality constraints satisfied. By definition of elementary contexts, there are two descendant nodes $\alpha_1 \lhd^+ \beta_1$ contained in $\mathrm{cxt}_t(u_1)$, and a state $q \in Q - (\mathrm{dom}(=_A) \cup \mathrm{dom}(\neq_A))$ such that $r(\alpha_1) = r(\beta_1) = q$. For all $i \in \{2, \ldots, n\}$, we define α_i as the node below u_i such that $\mathrm{path}_t(u_i, \alpha_i) \equiv \mathrm{path}_t(u_1, \alpha_1)$ (it exists since $u_1 \sim_{t,r} u_i$ and $t|_{u_1} = t|_{u_i}$ by Prop. 8). Note that by Prop. 8, $\alpha_1 \sim_{t,r} \alpha_i$. We define the nodes β_i similarly, and also get $\beta_1 \sim_{t,r} \beta_i$. By condition 2, for all $i \in \{1, \ldots, n\}$, $r(\alpha_i) = r(\beta_i) = q$. Hence there is a unary context C over Σ such that for all $i \in \{1, \ldots, n\}$, $t|_{\alpha_i} = C[t|_{\beta_i}]$. We let $t^1 = t$, $t^2 = t\{\alpha_i \leftarrow C[C[t|_{\beta_i}]],\ i = 1, \ldots, n\}$ the tree t where the subtree at node α_i has been substituted by $C[C[t|_{\beta_i}]]$, for all $i \in \{1, \ldots, n\}$. Similarly, for all $j \in \mathbb{N}$, we define t^j by iterating this substitution j times. We define r^j similarly. First note that r^j is a run of A on t^j. Moreover, since we make the substitution in parallel at isomorphic positions in the elementary contexts rooted at $[u_1]_{t,r}$, and by definition of $\sim_{t,r}$, the equality constraints are still satisfied by r^j on t^j, for all $j \in \mathbb{N}$. This pumping is illustrated in Fig. 7.

Existence of a Repairing Seed We let $\uparrow\omega$ be the ancestors of the nodes of ω, *i.e.* the set $\{w \mid \exists w' \in \omega,\ w \lhd^* w'\}$. The pumpings r^i, t^i $(i \in \mathbb{N})$ may create unsatisfied disequality constraints which involve a node (or two) of $\uparrow\omega$. On the contrary, it may also repair unsatisfied disequality constraints in r and t. We define a set of pairs of nodes $(u, v) \in N_t \times N_t$ between which there is an unsatisfied

constraint in t and r, or which may create a disequality constraint when pumping, but for which there is some $i \in \mathbb{N}$ such that the constraint is satisfied in t^i and r^i. These pairs are called candidates. We denote by $\mathrm{Cand}(\omega)$ this set. For all $u, v \in N_t$, $(u, v) \in \mathrm{Cand}(\omega)$ if (i) $u \in \uparrow\omega$, (ii) $r(u) \neq_A r(v)$, and (iii) if $t|_u = t|_v$, then there is $w \in \omega$ such that $\mathrm{Rep}(t, r, u, v, w)$ holds.

Note that by definition of the pumping, for all $i > 0$, for all nodes $u \in N_t$ such that $r(u) \in \mathrm{dom}(=_A) \cup \mathrm{dom}(\neq_A)$, u is still a node of t^i. For all $i > 0$, and all pairs $(u, v) \in \mathrm{Cand}(\omega)$, we say that i is *incompatible* with (u, v) if $t^i|_u = t^i|_v$.

Claim 1*. For all pairs $(u, v) \in \mathrm{Cand}(\omega)$, there is at most one $i > 0$ such that i is incompatible with (u, v).*

It is proved at the end of this proof. From *Claim 1*, we deduce that there is $i_0 \in \mathbb{N}$ such that i_0 is compatible with all pairs of $\mathrm{Cand}(\omega)$. We let $t' = t^{i_0}$ and $r' = r^{i_0}$.

Remark 1 By definition of the pumping, all inequality constraints between pairs of nodes $(u, v) \in \mathrm{Cand}(\omega)$ are satisfied in r'.

Correctness We prove that (t', r') is a seed. By definition of the pumping, condition 2 of the definition of a seed still holds for $\sim_{t',r'}$, since we pump the elementary contexts in parallel below all nodes of ω, both in r and t. Moreover, the equality constraints are still satisfied (thanks to parallel pumping), the root of r' is labeled by a final state, and the pumping does not increase the number of nodes ordered by $\lhd^*$ which are labeled in $\mathrm{dom}(\neq_A)$. Hence condition 1 of the definition of a seed holds for r'. To prove that (t', r') satisfies condition 3 of the seed definition, let $v_1, v_2 \in N_{t'}$ such that $t'|_{v_1} = t'|_{v_2}$ and $r'(v_1) \neq_A r'(v_2)$. We need to consider only two cases:

Case 1. $v_1 \notin \uparrow\omega$ and $v_2 \notin \uparrow\omega$. Since $r(v_1), r(v_2) \in \mathrm{dom}(\neq_A)$, neither v_1 nor v_2 are in the elementary contexts rooted at the nodes of ω. Hence the subtrees at positions v_1 and v_2 have not changed during the pumping. Hence $t|_{v_1} = t|_{v_2}$. Since (t, r) is a seed, there is an equivalence class $c \in \mathcal{C}(r)$ and $u \in c$ such that $\mathrm{Rep}(t, r, v_1, v_2, u)$ holds. Necessarily, $c \subseteq \mathcal{C}(r')$, otherwise it would mean that $u \in \omega$, which would contradict $v_1 \notin \uparrow\omega$ or would contradict $v_2 \notin \uparrow\omega$. Thus $\mathrm{Rep}(t', r', v_1, v_2, u)$ holds;

Case 2 $v_1 \in \uparrow\omega$. In this case $(v_1, v_2) \notin \mathrm{Cand}(\omega)$, otherwise $t'|_{v_1} \neq t'|_{v_2}$. By definition of $\mathrm{Cand}(\omega)$, $t|_{v_1} = t|_{v_2}$ and for all $w \in \omega$, $\mathrm{Rep}(t, r, v_1, v_2, w)$ does not hold. Since (t, r) is a seed, there is are an equivalence class $c \subseteq \mathcal{C}(r)$ and $u \in c$ such that $\mathrm{Rep}(t, r, v_1, v_2, u)$ holds. Hence $u \notin \omega$, and $c \neq \omega$. Therefore $c \subseteq \mathcal{C}(r')$ and $\mathrm{Rep}(t', r', v_1, v_2, u)$ holds.

$\deg(t', r') < \deg(t, r)$. Let $(u_1, u_2) \in N_{t'} \times N_{t'}$ be an unsatisfied inequality constraints in r'. We prove that it was also an unsatisfied inequality constraints in r. Suppose the contrary, *i.e.* $t|_{u_1} \neq t_{u_2}$. This means that either u_1 or u_2 (or both) is above some node u of ω (otherwise we would have $t|_{u_1} = t'_{u_1}$ and $t|_{u_2} = t'|_{u_2}$). By definition of $\mathrm{Cand}(\omega)$, we get $(u_1, u_2) \in \mathrm{Cand}(\omega)$, which contradicts Remark 1.
End of proof of Lemma 20 □

Proof of Claim 1 Suppose that there are two indices $i < j$ incompatible with (u, v). We consider the following two cases:

Case 1 $v \notin \uparrow\omega$. By hypothesis, $r^i(v) \neq_A r^i(u)$, $t^i|_v = t^i|_u$, $r^j(v) \neq_A r^j(u)$ and $t^j|_v = t^j|_u$. Since $r(v) \in \mathrm{dom}(\neq_A)$, and $v \notin \uparrow\omega$, v is below, or incomparable, to any node which belongs to an elementary context rooted at a node of ω. Hence the subtree at node v remains unchanged during the pumping. Therefore $t|_v = t^i|_v = t^j|_v$. Since by hypothesis, $t^i|_v = t^i|_u$ and $t^j|_v = t^j|_u$, we also get $t^j|_u = t^i|_u$. Since $i < j$, and $u \in \uparrow\omega$, by definition of the pumping, we have $\|t^j|_u\| > \|t^i|_u\|$, which contradicts $t^j|_u = t^i|_u$;

Case 2 $v \in \uparrow\omega$. Again we consider two cases:

Case 2.1 $t|_u = t|_v$. By definition of candidates, there is $u' \in \omega$ such that $\mathrm{Rep}(t, r, u, v, u')$ holds. Suppose that $u \lhd^* u'$ and let v' be the node below v such that $\mathrm{path}_t(u, u') \equiv \mathrm{path}_t(v, v')$ (it exists since $t|_u = t|_v$). By definition of the predicate Rep, $u' \not\sim_{t,r} v'$. Since $\mathrm{path}_t(u, u') \equiv \mathrm{path}_t(v, v')$, and $t|_u = t|_v$, we also get $t|_{u'} = t|_{v'}$. This implies that there is no node of ω comparable to v' (by $\lhd^*$). Therefore the subtree at position v' does not change during the pumping. In particular, $t^i|_{v'} = t^j|_{v'}$. Since $u' \in \omega$, we pump below u'. Thus we have $t^i|_{u'} \neq t^j|_{u'}$. Therefore, either $t^i|_{u'} \neq t^i|_{v'}$ or $t^j|_{u'} \neq t^j|_{v'}$. Since $\mathrm{path}_t(u, u') \equiv \mathrm{path}_t(v, v')$, and there is no node of ω comparable to v', by definition of the pumping, we also have $\mathrm{path}_{t^i}(u, u') \equiv \mathrm{path}_{t^i}(v, v')$ and $\mathrm{path}_{t^j}(u, u') \equiv \mathrm{path}_{t^j}(v, v')$. Therefore, either $t^i|_v \neq t^i|_u$ or $t^j|_v \neq t^j|_u$;

Case 2.2 $t|_u \neq t|_v$. By hypothesis, $t^i|_u = t^i|_v$. We first prove that there is necessarily $u' \in \omega$ such that $\mathrm{Rep}(t^i, r^i, u, v, u')$ holds. Suppose the contrary. This means that for all $u' \in \omega$ such that $u \lhd^* u'$, and for all v' such that $\mathrm{path}_{t^i}(u, u') \equiv \mathrm{path}_{t^i}(v, v')$, we have $v' \sim_{t,r} u'$. And symmetrically, for all $v' \in \omega$ such that $v \lhd^* v'$, and for all u' such that $\mathrm{path}_{t^i}(u, u') \equiv \mathrm{path}_{t^i}(v, v')$, we have $u' \sim_{t,r} v'$. It is not difficult to see that in that case, we already had $t|_u = t|_v$. Informally, we can pump backward to get (t^{i-1}, r^{i-1}) from (t^i, r^i), with the same technique. This would preserve the equality $t^{i-1}|_u = t^{i-1}|_v$. Applying this inductively we can get (t, r) and in particular $t|_u = t|_v$, which is absurd. Therefore there is $u' \in \omega$ such that $\mathrm{Rep}(t^i, r^i, u, v, u')$. Suppose that $u \lhd^* u'$ (the case $v \lhd^* u'$ is symmetric). Let v' such that $v \lhd^* v'$ and $\mathrm{path}_{t^i}(v, v') \equiv \mathrm{path}_{t^i}(u, u')$. By definition of Rep, we have $u' \not\sim_{t^i, r^i} v'$. Since $t^i|_u = t^i|_v$, we also get $t^i|_{u'} = t^i|_{v'}$. For the same reasons as the previous case, there is no node $w \in \omega$ which is comparable to v'. Since we pump only in the elementary contexts rooted at nodes of ω, the subtree rooted at v' does not change during pumping. Since the subtree rooted at u' changes during pumping and $\mathrm{path}_{t^i}(u, u') \equiv \mathrm{path}_{t^i}(v, v')$, we necessarily have $t^j|_u \neq t^j|_v$, which contradicts the hypothesis. □

Lemma 21. *Let (A, k) be a rigid vbTAGED and $B = 2(k + |dom(=_A)|)|Q|$. If $\mathcal{L}(A, k) \neq \varnothing$, there is a seed (t, r) such that $height(t) \leq B$.*

Before proving this lemma, let us informally explain the main ideas. Given a tree t and a successful run r on it (this is a particular case of seed), we decrease the size of elementary contexts rooted at equivalent nodes in parallel, while preserving

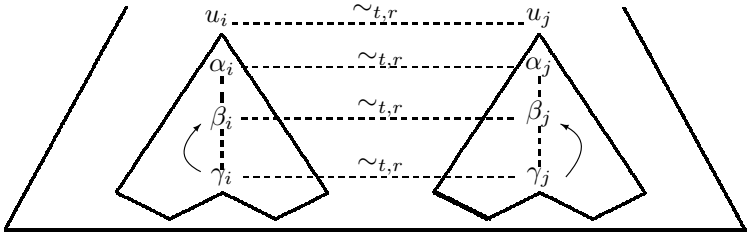

Fig. 8: Decreasing the size of elementary contexts

the equality constraints. This is done by collapsing the loops they contain. It may "break" some inequality constraints, but we do it in a way that preserves the fact to be a seed.

Given an elementary context C in r, we say that C contains a 3-loop if there is a chain of nodes of C which contains at least 3 nodes labeled by the same state. For the proof of the lemma we will need two claims. More exactly:

Claim 1 *For all seeds (t, r) such that there is an elementary context of r which contains a 3-loop, there exists a seed (t', r') such that $|N_{t'}| < |N_t|$*

Proof. The proof of this claim is divided in several parts. We first show how to construct t' and r' and then prove the correctness. We pump in parallel in elementary contexts which contain a 3-loop (and that are rooted to equivalent nodes). This ensures that equality constraints remain satisfied. In particular, in each elementary context, we collapse the two greatest nodes of the 3-loop (for $\lhd^*$), so that there is still a loop in the elementary context after pumping.

By hypothesis, there is $u \in \mathcal{C}(r)$ such that $\mathrm{cxt}_r(u)$ contains a 3-loop. By definition of a seed, $r|_v = r|_u$, for all $v \in [u]_{t,r}$. Thus $v \in \mathcal{C}(r)$ for all $v \in [u]_{t,r}$. By definition of $\sim_{t,r}$, all the nodes of $[u]_{t,r}$ are incomparable. Let $n = |[u]_{t,r}|$ and $\{u_1, \ldots, u_n\} = [u]_{t,r}$. Let C be the n-ary context over Q such that $r = C[r|_{u_1}, \ldots, r|_{u_n}]$. For all $i \in \{1, \ldots, n\}$, there are three nodes $\alpha_i, \beta_i, \gamma_i \in N_{\mathrm{cxt}_r(u_i)}$ and a state $q \notin \mathrm{dom}(=_A) \cup \mathrm{dom}(\neq_A)$ such that $\alpha_i \lhd^+ \beta_i \lhd^+ \gamma_i$, and $r(\alpha_i) = r(\beta_i) = r(\gamma_i) = q$. Moreover, we can take $\alpha_i, \beta_i, \gamma_i$ such that for all $i, j \in \{1, \ldots, n\}$, we have $\mathrm{path}_t(u_i, \alpha_i) \equiv \mathrm{path}_t(u_j, \alpha_j)$, $\mathrm{path}_t(\alpha_i, \beta_i) \equiv \mathrm{path}_t(\alpha_j, \beta_j)$, and $\mathrm{path}_t(\beta_i, \gamma_i) \equiv \mathrm{path}_t(\beta_j, \gamma_j)$. We let $r' = C[r_1, \ldots, r_n]$, where for all $i \in \{1, \ldots, n\}$, r_i is the tree $r|_{u_i}$ in which the subtree rooted at β_i has been substituted by $r|_{\gamma_i}$. We do the corresponding substitution in t and obtain a tree t'. This pumping is described in Fig. 8.

It is technical but not difficult to prove that conditions 1 and 2 of the definition of a seed hold for (t', r'). We now prove that condition 3 also holds. Let $(v_1, v_2) \in \mathrm{cst}^A_{\neq}(t', r')$ such that $t'|_{v_1} = t'|_{v_2}$. We consider two cases:

Case 1 $t|_{v_1} \neq t|_{v_2}$. This means that the pumping has "broken" this disequality. Let $[u]_{t,r} = \{u_1, \ldots, u_n\}$ be the nodes defined in the definition of the pumping,

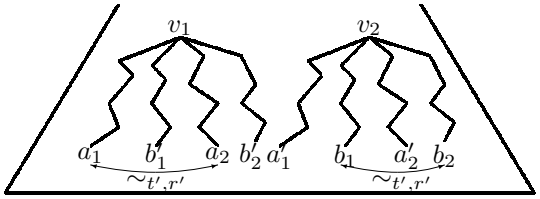

Fig. 9:

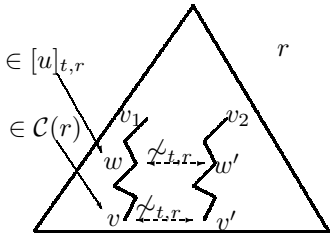

Fig. 10:

i.e. the nodes which enroot elementary contexts in which we have pumped. Note that we still have $[u]_{t,r} \subseteq N_{t'}$ and $[u]_{t,r} = [u]_{t',r'}$ since we have pumped below the nodes of $[u]_{t,r}$. Let $\{a_1, \ldots, a_{n_1}\} \subseteq [u]_{t,r}$ (resp. $\{b_1, \ldots, b_{n_2}\} \subseteq [u]_{t,r}$) the nodes of $[u]_{t,r}$ which are below v_1 (resp. v_2). For all $\ell \in \{1, \ldots, n_1\}$, let $a'_\ell \in N_{t'}$ such that $\mathrm{path}_{t'}(v_2, a'_\ell) \equiv \mathrm{path}_{t'}(v_1, \alpha_\ell)$ (it exists since $t'|_{v_1} = t'|_{v_2}$). Similarly, for all $\ell \in \{1, \ldots, n_2\}$, let b'_ℓ the node below v_1 such that $\mathrm{path}_{t'}(v_1, b'_\ell) \equiv \mathrm{path}_{t'}(v_2, b_\ell)$ (it is depicted in Fig. 9).We prove that there is a node w in $\{a_1, \ldots, a_{n_1}, b_1, \ldots, b_{n_2}\}$ such that $\mathrm{Rep}(t', r', v_1, v_2, w)$ holds. Suppose that for all $\ell \in \{1, \ldots, n_1\}$, we have $a_\ell \sim_{t',r'} a'_\ell$, and for all $\ell \in \{1, \ldots, n_2\}$, we have $b_\ell \sim_{t',r'} b'_\ell$. By definition of the pumping, the maximal context of t which have nodes of $[u]_{t,r}$ as holes has not changed during pumping. Intuitively this means that we have pumped similarly at isomorphic positions, so that the two trees $t|_{v_1}$ and $t|_{v_2}$ were already equal, which contradicts $t|_{v_1} \neq t|_{v_2}$. Thus there is $\ell_1 \in \{1, \ldots, n_1\}$ or $\ell_2 \in \{1, \ldots, n_2\}$ such that $a_{\ell_1} \not\sim_{t',r'} a'_{\ell_1}$ or $b_{\ell_2} \not\sim_{t',r'} b'_{\ell_2}$, which concludes this case.

Case 2 $t|_{v_1} = t|_{v_2}$. By definition of a seed, there is $u \in \mathcal{C}(r)$ and $v \sim_{t,r} u$ such that $\mathrm{Rep}(t, r, v_1, v_2, v)$ holds. Suppose that $v_1 \lhd^* v$, the other case being symmetric. Let v' be such that $\mathrm{path}_t(v_1, v) \equiv \mathrm{path}_t(v_2, v')$. By definition of the predicate Rep, $v \not\sim_{t,r} v'$. We now consider three cases:
(i) there is a node $w \in [u]_{t,r}$ such that $v_1 \lhd^* w \lhd^* v$. Let w' such that $\mathrm{path}_t(v_1, w) \equiv \mathrm{path}_t(v_2, w')$. This situation is depicted in Fig. 10. If $w \sim_{t,r} w'$, then by Proposition 8, we also have $v \sim_{t,r} v'$, which is impossible. Hence $w \not\sim_{t,r} w'$, and since we pump only below nodes of $[u]_{t,r}$, w is still a node of t' and $w \in \mathcal{C}(r)$. Therefore $\mathrm{Rep}(t', r', v_1, v_2, w)$ holds;
(ii) there is a node $w \in [u]_{t,r}$ such that $v_2 \lhd^* w \lhd^* v'$. With exactly the same arguments we can prove that $\mathrm{Rep}(t', r', v_1, v_2, w)$ holds;
(iii) there is no node $w \in [u]_{t,r}$ such that $v_1 \lhd^* w \lhd^* v$ or $v_2 \lhd^* w \lhd^* v'$. Since we pump inside elementary contexts rooted at nodes of $[u]_{t,r}$, and $r(v_1), r(v_2) \in \mathrm{dom}(\neq_A)$, by definition of elementary contexts, neither v_1 nor v_2 can be a node of some context $\mathrm{cxt}_t(v)$, for $v \in [u]_{t,r}$. By definition of the pumping, $\mathrm{path}_{t'}(v_1, v)$ is still edge-isomorphic to $\mathrm{path}_{t'}(v_2, v')$. Moreover, $v \in \mathcal{C}(r')$. Finally, thanks to Claim 2 (next stated), since $v \not\sim_{t,r} v'$, we also get $v \not\sim_{t',r'} v'$. Therefore $\mathrm{Rep}(t', r', v_1, v_2, v)$ holds. □

Claim 2. *$N_{t'} \subseteq N_t$ and for all $v_1, v_2 \in N_{t'}$, $v_1 \sim_{t',r'} v_2$ iff $v_1 \sim_{t,r} v_2$.*

Proof. Let $v_1, v_2 \in N_{t'}$ such that $v_1 \sim_{t,r} v_2$ and $v_1 \neq v_2$ (the case $v_1 = v_2$ is obvious). We can prove[f] that there are $w_1, w_2 \in N_t$ such that $w_1 \lhd^* v_1$, $w_2 \lhd^* v_2$ and $r(w_1) =_A r(w_2)$. Suppose that $w_1 \notin N_{t'}$ or $w_2 \notin N_{t'}$. This means that w_1, w_2 have been removed by the pumping. Since the elementary contexts in which we pump do not contain nodes labeled by states from $\mathrm{dom}(=_A)$ (except at their root), this means that w_1 and w_2 are below elementary contexts, hence their whole subtrees have been removed by the pumping. In particular, v_1 and v_2 are removed, which is impossible. By contradiction, $w_1, w_2 \in N_{t'}$. We consider two cases:

Case 1: there is no node $u' \in [u]_{t,r}$ such that $w_1 \lhd^* u' \lhd^* v_1$ or $w_2 \lhd^* u' \lhd^* v_2$. Then $\mathrm{path}_{t'}(w_1, v_1) \equiv \mathrm{path}_{t'}(w_2, v_2)$. Since $r'(w_1) = r(w_1)$ and $r'(w_2) = r(w_2)$, we get $r'(w_1) =_A r'(w_2)$ and therefore $w_1 \sim_{t',r'} w_2$, from which we get $v_1 \sim_{t',r'} v_2$.

Case 2: there is $u' \in [u]_{t,r}$ such that $w_1 \lhd^* u' \lhd^* v_1$. Let $u'' \in N_t$ such that $w_2 \lhd^* u'' \lhd^* v_2$ and $\mathrm{path}_t(w_1, u) \equiv \mathrm{path}_t(w_2, u')$. Since $w_1 \sim_{t,r} w_2$, we also have $u'' \sim_{t,r} u'$. Hence $u'' \in [u]_{t,r}$. Since we pump in parallel in $\mathrm{cxt}_t(u')$ and $\mathrm{cxt}_t(u'')$, $\mathrm{path}_{t'}(w_1, v_1)$ is still edge-isomorphic to $\mathrm{path}_{t'}(w_2, v_2)$, so that $v_1 \sim_{t',r'} v_2$.

Conversely, suppose that $v_1 \sim_{t',r'} v_2$ and $v_1 \neq v_2$ (the case $v_1 = v_2$ is obvious). For the same reason as before, there are $w_1, w_2 \in N_{t'}$ such that $w_1 \lhd^* v_1$, $w_2 \lhd^* v_2$ and $r'(w_1) =_A r'(w_2)$. Since $N_{t'} \subseteq N_t$, we necessarily have $w_1, w_2, v_1, v_2 \in N_t$. By definition of the pumping, we also have $w_1 \lhd^* v_1$ and $w_2 \lhd^* v_2$. Suppose that $\mathrm{path}_t(w_1, v_1)$ is not edge-isomorphic to $\mathrm{path}_t(w_2, v_2)$. We show a contradiction (hence this will prove $v_1 \sim_{t,r} v_2$). Since $\mathrm{path}_t(w_1, v_1)$ and $\mathrm{path}_t(w_2, v_2)$ are not edge-isomorphic, and after pumping $\mathrm{path}_{t'}(w_1, v_1)$ and $\mathrm{path}_{t'}(w_2, v_2)$ are isomorphic, necessarily a pumping has occurred in an elementary context rooted at some node $u' \in [u]_{t,r}$ such that $w_1 \lhd^* u' \lhd^* v_1$ or $w_2 \lhd^* u' \lhd^* v_2$. Suppose that $w_1 \lhd^* u'$, and let u'' such that $\mathrm{path}_t(w_1, u') \equiv \mathrm{path}_t(w_2, u'')$ (it exists since $w_1 \sim_{t,r} w_2$ and $t|_{w_1} = t|_{w_2}$). If u'' does not belong to the path from w_2 to v_2, then after pumping, we would still have $\mathrm{path}_{t'}(w_1, v_1)$ non-isomorphic to $\mathrm{path}_{t'}(w_2, v_2)$, since we pump below u' and u''. Thus u'' belongs to $\mathrm{path}_t(w_2, v_2)$. It is not difficult to show that since we pump in parallel at equivalent positions, we still have $\mathrm{path}_{t'}(w_1, v_1)$ non-isomorphic to $\mathrm{path}_{t'}(w_2, v_2)$, which is a contradiction. □

Proof of Lemma 21 If $\mathcal{L}(A, k) \neq \varnothing$, there is $t \in \mathcal{L}(A, k)$ and r a successful run of (A, k) on t. By Lemma 12, we can suppose that for all nodes $u, v \in N_t$, if $u \sim_{t,r} v$, then $r|_u = r|_v$. Note that (t, r) is a seed. The idea is to apply exhaustively

[f]It can be shown by induction: it is obvious if $v_1 = v_2$ or $v_1 \leftrightarrow_{t,r} v_2$, by definition of $\leftrightarrow_{t,r}$. Suppose that there is v_3 such that $v_1 \sim_{t,r} v_3$ and $v_3 \leftrightarrow_{t,r} v_2$, and there are some nodes w_1, w_3 above v_1, v_3 such that $r(w_3) =_A r(w_1)$. By definition of $\leftrightarrow_{t,r}$, there are w_3' and w_2' such that $\mathrm{path}_t(w_3', v_3) \equiv \mathrm{path}_t(w_2', v_2)$, and $r(w_3') =_A r(w_2')$. By definition of $\sim_{t,r}$, we have $w_3' \sim_{t,r} w_2'$, and by definition of a seed, we get $r|_{w_3'} = r|_{w_2'}$, and, similarly, we get $r|_{w_1} = r|_{w_3}$. Suppose that $w_3 \lhd^* w_3'$, and let w_1' above v_1 such that $\mathrm{path}_t(w_1, w_1') \equiv \mathrm{path}_t(w_3, w_3')$: it exists since $r|_{w_1} = r|_{w_3}$, moreover, $r(w_1') =_A r(w_3')$. Since $r(w_3') =_A r(w_2')$ and $=_A \subseteq id_Q$, we also have $r(w_1') =_A r(w_2')$. The case $w_3' \lhd^+ w_3$ is proved similarly.

Claim 1 until we can bound the height of a seed by some constant that depends only on A and k.

Suppose that the height of t is strictly greater than B. In any $\lhd^*$-ordered chain, they are at most k nodes labeled by a state of $\mathrm{dom}(\neq_A)$ in r, and at most $|\mathrm{dom}(=_A)|$ nodes labeled by a state of $\mathrm{dom}(=_A)$ (otherwise there would be two different descendant nodes enrooting equal subtrees in t, which is impossible). Let π be a root-to-leaf path of r of length strictly greater than B. If there is no node in π labeled by a state of $\mathrm{dom}(\neq_A) \cup \mathrm{dom}(=_A)$, then there are necessarily a loop in π that we can collapse to create a new tree and a new run of strictly less size which still form a seed. Otherwise let u be the least node in π labeled by a state of $\mathrm{dom}(\neq_A) \cup \mathrm{dom}(=_A)$. We now consider two cases. If the path from the root to u is strictly longer than B, again this means that there is a loop that we can collapse without changing the constraints. Otherwise, by definition of B, and the fact that there are at most $|\mathrm{dom}(=_A)|$ nodes labeled by an equality state in π, there is necessarily a node in π which enroots an elementary context which contains a 3-loop. Applying Claim 1 allows us to get a seed (t', r') of strictly less size than (t, r).

Finally, we can iterate this reasoning until there is no root-to-leaf path of length strictly greater than B, and we are done. □

Theorem 22. *Let (A, k) be a vbTAGED. Testing emptiness of (A, k) can be done in 2NEXPTIME. It is in NEXPTIME if A is rigid and $k \leq |Q|$ (or k is represented by a unary encoding).*

Proof. Let $B = 2(k + |\mathrm{dom}(=_A)|)|Q|$ if A is rigid, and $B = (k + 2^{|Q|})2^{|Q|+1}$ otherwise. Let A' equal to A if A is rigid, or an equivalent rigid TAGED otherwise (as constructed in Corollary 11). By Lemmata 21, 19 and 20, $L(A, k) \neq \varnothing$ iff there exists a seed (t, r) (for A') such that $\mathit{height}(t) \leq B$. Therefore, it suffices to guess a seed (t, r) of height at most B (and sizes at most 2^{B+1}). Moreover, checking whether a pair (t, r) is a seed can be done in PTIME in $\|t\|$, $\|r\|$, and $\|A'\|$. If $B = (k + 2^{|Q|})2^{|Q|+1}$, this gives a non-deterministic algorithm doubly exponential in $|Q|$ and $\|k\|$, since $\|k\| = log_2(k)$. However, if $B = 2(k + |\mathrm{dom}(=_A)|)|Q|$ and the encoding of k is unary, or k is polynomial in $|Q|$, the non-deterministic algorithm is simply exponential in $|Q|$ and $\|k\|$. □

Acknowledgments We are grateful to the referees for their valuable comments. We warmly thank Hitoshi Osaki and Frédéric Servais for fruitful discussions.

References

[1] A. Aiken, D. Kozen, and E. L. Wimmers. Decidability of systems of set constraints with negative constraints. *Information and Computation*, 122(1):30–44, 1995.

[2] S. Anantharaman, P. Narendran, and M. Rusinowitch. Closure properties and decision problems of dag automata. *Information Processing Letters*, 94(5):231–240, 2005.

[3] L. Barguñó, C. Creus, G. Godoy, F. Jacquemard, and C. Vacher. The emptiness problem for tree automata with global constraints. In *IEEE Symposium on Logic in Computer Science (LICS)*, 2010. To appear.

[4] B. Bogaert and S. Tison. Equality and disequality constraints on direct subterms in tree automata. In *9th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 577 of *LNCS*, pages 161–171, 1992.

[5] L. Cardelli and G. Ghelli. TQL: A Query Language for Semistructured Data Based on the Ambient Logic. *Mathematical Structures in Computer Science*, 14:285–327, 2004.

[6] W. Charatonik. Automata on dag representations of finite trees. Technical report, 1999.

[7] W. Charatonik and L. Pacholski. Set constraints with projections are in NEXPTIME. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 1994.

[8] H. Comon and V. Cortier. Tree automata with one memory, set constraints and cryptographic protocols. *Theoretical Computer Science (TCS)*, 331(1):143–214, 2005.

[9] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: http://www.grappa.univ-lille3.fr/tata, 2007.

[10] M. Dauchet, A.-C. Caron, and J.-L. Coquidé. Reduction properties and automata with constraints. *Journal of Symbolic Computation (JSC)*, 20:215–233, 1995.

[11] E. Filiot. *Logics for n-ary queries in trees*. PhD thesis, Lille 1 University, 2008.

[12] E. Filiot, J.-M. Talbot, and S. Tison. Satisfiability of a spatial logic with tree variables. In *Computer Science Logic (CSL)*, pages 130–145, 2007.

[13] E. Filiot, J.-M. Talbot, and S. Tison. Tree automata with global constraints. In *12th International Conference on Developments in Language Theory (DLT)*, Lecture Notes in Computer Science. Springer Verlag, 2008.

[14] R. Gilleron, S. Tison, and M. Tommasi. Some new decidability results on positive and negative set constraints. In *Proceedings of the First International Conference on Constraints in Computational Logics (CCL)*, pages 336–351, 1994.

[15] V. Halava, M. Hirvensalo, and R. de Wolf. Marked PCP is decidable. *Theoretical Computer Science (TCS)*, 255(1-2):193–204, 2001.

[16] F. Jacquemard, F. Klay, and C. Vacher. Rigid tree automata. In *Proceedings of the 3rd International Conference on Language and Automata Theory and Applications (LATA'09)*, volume 5457, pages 446–457. Springer, 2009.

[17] F. Jacquemard, M. Rusinowitch, and L. Vigneron. Tree automata with equality constraints modulo equational theories. Journal of Logic and Algebraic Programming 75(2): 182-208, 2008.

[18] W. Karianto and C. Löding. Unranked tree automata with sibling equalities and disequalities. In *International Colloquium on Automata Languages and Programming (ICALP)*, volume 4596, pages 875–887. Springer, 2007.

[19] J. Mongy. *Transformation de noyaux reconnaissables d'arbres. Forêts RATEG*. PhD thesis, Université de Lille, 1981.

[20] F. Neven and T. Schwentick. Xpath containment in the presence of disjunction, dtds, and variables. In *International Conference on Database Theory (ICDT)*, pages 315–329, London, UK, 2002. Springer-Verlag.

[21] T. Schwentick. Automata for xml – a survey. *Journal Computer and System Science (JCSS)*, 73(3):289–315, 2007.

[22] K. Stefansson. Systems of set constraints with negative constraints are nexptime-complete. In *IEEE Symposium on Logic in Computer Science (LICS)*, 1994.

[23] J. W. Thatcher and J. B. Wright. Generalized finite automata theory with an ap-

plication to a decision problem of second-order logic. *Mathematical Systems Theory (MST)*, 2(1):57–81, 1968.