HAL open science

Tree Automata with Global Constraints

Emmanuel Filiot, Jean-Marc Talbot, Sophie Tison

To cite this version:

Emmanuel Filiot, Jean-Marc Talbot, Sophie Tison. Tree Automata with Global Constraints. 12th International Conference on Developments in Language Theory (DLT), Sep 2008, Kyoto, Japan. pp.314-326. inria-00292027v1

HAL Id: inria-00292027

https://inria.hal.science/inria-00292027v1

Submitted on 4 Nov 2008 (v1), last revised 8 Jun 2009 (v2)

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

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

Tree Automata with Global Constraints

Emmanuel Filiot[1] Jean-Marc Talbot[2] Sophie Tison[1]

[1] INRIA Lille - Nord Europe, Mostrare Project, University of Lille 1 (LIFL, UMR 8022 of CNRS)
[2] University of Provence (LIF, UMR 6166 of CNRS), Marseille

Abstract. A tree automaton with global tree equality and disequality constraints, TAGED for short, is an automaton on trees which allows to test (dis)equalities between subtrees which may be arbitrarily faraway. In particular, it is equipped with an (dis)equality 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 decidability of emptiness of several classes. We give two applications of TAGEDs: decidability of an extension of Monadic Second Order Logic with tree isomorphism tests and of unification with membership constraints. These results significantly improve the results of [10].

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 [3, 9, 13, 14]. In particular, the transitions of these tree automata are fired as soon as the subtrees of the current tree satisfy some structural (dis)equality. 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 need global constraints. For instance, it might be useful 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 occurs 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 allow to test tree disequalities),

Boolean operations, and regular constraints on variables. In [10], we study the spatial logic TQL, which in particular, allows to define such patterns. We proved decidability of a powerful fragment of this logic, by reduction to emptiness test of a new class of tree automata, called *Tree Automata with Global Equalities and Disequalities* (TAGEDs for short). These are tree automata A equipped with an equality $=_A$ and a disequality $\neq_A$ relation 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 whenever two subtrees of t evaluate to two states q_1, q_2 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 on their own, since they are somehow orthogonal to usual automata with constraints [3]. 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 are successfully tested to be equal), 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.
The main result of [10] was decidability of emptiness of a subclass of TAGEDs, called bounded TAGEDs, which allow only a bounded number (by some constant independent of the tree) of (dis)equality tests on the run. In this paper, we prove several properties of TAGED-definable languages (closure by union and intersection, non-closure by complement). We prove results on TAGEDs (non-determinization, undecidability of universality). The other main results are decidability of emptiness of several classes of TAGEDs which significantly improves the result of [10], and uses different and simpler techniques. In particular, we prove a pumping lemma for TAGEDs which performs a bounded number of disequality tests along paths (and arbitrarily many equality tests).
We give two applications of TAGEDs. The first is decidability of an extension of MSO with tree isomorphism tests. The second application concerns a first-order disunification problems, with (regular) membership constraints. Dealing with membership constraints has been done in several papers. In [8], the authors prove solvability of first-order formulas whose atoms are either equations between terms or membership constraints $t \in L$ where L is a regular tree language. In [16], the authors propose an algorithm to solve iterated matching of hedges against terms with flexible arity symbols, one-hole context and sequence variables constrained to range over a regular language. In this paper, we extend the logic of [8] with context variables (with arbitrarily many holes, and membership constraints) to allow arbitrary depth matching. Context unification is still an open problem, but motivated by XML tasks, we do not need to do full context unification. We prefer to impose a strong linearity condition on context variables. We prove that, even with this restriction, solvability of first-order formulas over these atoms is undecidable. We introduce an existential fragment for which satisfiability is decidable by reduction to emptiness of a class of TAGEDs.

Related Work 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 [18] 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 [9] whereas the second restricts tests to sibling subterms [3]. This latter class has recently been extended to unranked trees [14], the former one has been extended to equality modulo equational theories [13]. 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 [6]. Their emptiness problem is decidable, and they can simulate positive TAGEDs (TAGEDs performing only equality tests) which use at most one state per runs to test equalities. Finally, automata for DAGs are studied in [2, 4], they cannot be compared to positive TAGEDs, as they run on DAG representations of trees (with maximal sharing), and in TAGEDs, we cannot impose every equal subtrees to evaluate in the same state in a successful run.

We only sketch the proofs, but all the missing proofs are in the full paper version [24] and in the thesis [11].

Acknowledgements We are very grateful to Florent Jacquemard and Camille Vacher from the LSV (Cachan) for their valuable comments on the paper.

2 Trees and TAGED

Binary trees We start from a ranked alphabet Σ ranged over binary symbols f and constant symbols a. A *binary tree* t is a ground term over Σ. The set of binary trees over Σ is denoted by T_Σ. The set of *nodes* of a tree $t \in T_\Sigma$, denoted by N_t, is defined inductively as a set of words over $\{1, 2\}$ by: $N_a = \{\epsilon\}$ and $N_{f(t_1,t_2)} = \{\epsilon\} \cup \{\alpha.u \mid \alpha \in \{1, 2\},\ u \in N_{t_\alpha}\}$ (ϵ denotes the empty word and . the concatenation). For any tree t, and any node $u \in N_t$, we define the subtree at node u, denoted by $t|_u$, inductively by: $t|_\epsilon = t$, $f(t_1, t_2)|_{\alpha.u} = t_\alpha|_u$, $\alpha \in \{1, 2\}$. Note that we have $N_{t|_u} = \{v \mid u.v \in N_t\}$. We also denote by $O_t(u) \in \Sigma$ the label of node u in t. Finally, we denote by $\lhd$ the strict descendant relation between nodes. Hence, for all $u, v \in N_t$, $u \lhd v$ if u is a prefix of v (therefore the root is minimal for $\lhd$).

Tree Automata We define tree automata on binary trees, but the reader may refer to [7] 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 $O_t(u) \to O_r(u) \in \Delta$, and (iii) for all inner-nodes $u \in N_r$, we have $O_t(u)(O_r(u.1), O_r(u.2)) \to O_r(u) \in \Delta$.

A run r is *successful* if $O_r(\epsilon) \in F$. The *language recognized* (or defined) by A, denoted $L(A)$, is the set of trees t for which there exists a successful run of A. We consider binary and constant symbols only, but the two definitions above can be easily extended to symbols of other arity (in particular, we use unary symbols in several proofs and examples).

Example 1. Let Σ_b be the alphabet consisting of the two binary symbols $\wedge, \vee$, the unary symbol $\neg$, and the two constant symbols $0, 1$. Trees from T_{Σ_b} represents Boolean formulas. We define an automaton on Σ_b which accepts only Boolean formulas logically equivalent to 1. Its set of states (resp. final states) is defined by $Q_b = \{q_0, q_1\}$ (resp. $F_b = \{q_1\}$), and its set of rules Δ_b by, for all $b, b_1, b_2 \in \{0, 1\}$, and all $\oplus \in \{\wedge, \vee\}$:

$$b \to q_b \qquad \neg(q_b) \to q_{\neg b} \qquad \oplus(q_{b_1}, q_{b_2}) \to q_{b_1 \oplus b_2} \ .$$

Definition 1 (TAGED). *A TAGED is a 6-tuple $A = (\Sigma, Q, F, \Delta, =_A, \neq_A)$ such that:*
- *(Σ, Q, F, Δ) is a tree automaton;*
- *$=_A$ is a reflexive and symmetric binary relation on a subset of Q;*
- *$\neq_A$ is an irreflexive and symmetric binary relation on Q.*

A TAGED A is said to be positive (resp. negative) if $\neq_A$ (resp. $=_A$) is empty.

The notion of successful run differs from tree automata as we add equality and disequality constraints. A *run* r of the tree automaton (Σ, Q, F, Δ) on a tree t satisfies the equality constraints if $\forall u, v \in N_t,\ O_r(u) =_A O_r(v) \ \Rightarrow\ t|_u = t|_v$. Similarly, it satisfies the disequality constraints if $\forall u, v \in N_t, O_r(u) \neq_A O_r(v) \ \Rightarrow\ t|_u \neq t|_v$.
A run is *successful* (or *accepting*) if it is successful for the tree automaton (Σ, Q, F, Δ) and if it satisfies the constraints. The language accepted by A, denoted $L(A)$, is the set of trees t having a successful run for A. We denote by $\mathrm{dom}(=_A)$ the domain of $=_A$, i.e. $\{q \mid \exists q' \in Q, q =_A q'\}$. The set $\mathrm{dom}(\neq_A)$ is defined similarly. Finally, two TAGEDs are *equivalent* if they accept the same language.
In [10], we introduced the class of bounded TAGED, where in successful runs, the number of occurrences of states from $\mathrm{dom}(=_A) \cup \mathrm{dom}(\neq_A)$ is bounded by some fixed $k \in \mathbb{N}$. We proved emptiness to be decidable for that class. The classes we consider in this paper are either incomparable or strictly more expressive. All the results from Section 3 also hold for bounded TAGED. Note also that TAGED are strictly more expressive than tree automata, as illustrated by the next example.

Example 2. Let $Q = \{q, q_=, q_f\}$, $F = \{q_f\}$, and let Δ be defined as the set of following rules: $a \to q,\ a \to q_=,\ f(q, q) \to q,\ f(q, q) \to q_=,\ f(q_=, q_=) \to q_f$, for all $a, f \in \Sigma$. Let the positive TAGED $A_1 = (\Sigma, Q, F, \Delta, \{q_= =_{A_1} q_=\})$. Then $L(A_1)$ is the set $\{f(t, t) \mid f \in \Sigma, t \in T_\Sigma\}$, which is known to be non regular [7].

Example 3. Let $\mathcal{X}$ be a finite set of variables. We now define a TAGED A_{sat} which accepts tree representations of satisfiable Boolean formulas with free variables $\mathcal{X}$. We let $A_b = (\Sigma_b, Q_b, F_b, \Delta_b)$ be the automaton defined in Example 1. Every variable is viewed as a binary symbol, and we let $\Sigma_{\mathcal{X}} = \Sigma_b \cup \mathcal{X}$. 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 $\vee(t_1, t_2)$, where $t_1 = \wedge(x(0,1),\ y(0,1))$ and $t_2 = \neg(x(0,1))$. Now, we let $Q = Q_b \cup \{q_x \mid x \in \mathcal{X}\} \cup \{p_0, p_1\}$, for two fresh states p_0, p_1, and $F = F_b$. The idea is to choose non-deterministically to evaluate the leaf 0 or 1 below x to q_x, but not both, for all $x \in \mathcal{X}$. This means that we affect 0 or 1 to a particular occurrence of x. Then, by imposing that every leaf evaluated to q_x are equal,we can ensure that we have chosen to same Boolean value for all occurrences of x, for all $x \in \mathcal{X}$. This can be done with the set of rules Δ_b extended with the following rules, for all $b \in \{0,1\}$ and all $x \in \mathcal{X}$:

$$b \to p_b \qquad b \to q_x \qquad x(p_0, q_x) \to q_1 \qquad x(q_x, p_1) \to q_0.$$

Finally, for all $x \in \mathcal{X}$, we let $q_x =_{A_{sat}} q_x$.

The *(uniform) membership problem* is "given a TAGED A, given a tree t, does t belong to $L(A)$?". We can prove the following:

Proposition 1. *Membership is NP-complete for TAGED.*

Proof. Example 3 gives a polynomial reduction of SAT to membership of TAGEDs. To show it is in NP, it suffices to guess a labeling of the nodes of the tree by states, and then to verify that it is a run, and that equality and disequality constraints are satisfied. This can be done in linear time both in the size of the automaton and of the tree. □

3 Closure Properties of TAGEDs and Decision Problems

In this section, we prove closure properties of TAGED-definable languages.

Proposition 2 (Closure by union and intersection). *TAGED-definable languages are closed by union and intersection.*

Proof. Let $A = (\Lambda, Q, F, \Delta, =_A, \neq_A)$ and $A' = (\Lambda, Q', F', \Delta', =_{A'}, \neq_{A'})$ be two TAGEDs. Wlog, we suppose that $Q \cap Q' = \emptyset$. A TAGED accepting $L(A) \cup L(A')$ is defined by $A \cup A' = (\Lambda, Q \cup Q', F \cup F', \Delta \cup \Delta', =_A \cup =_{A'}, \neq_A \cup \neq_{A'})$.
For the closure by intersection, we use the usual product construction $A \times A'$ [7], whose set of final states if $F \times F'$. State equality $=_{A \times A'}$ is defined by $\{((q,q'),(p,p')) \mid q =_A p \text{ or } q =_{A'} p\}$, while $\neq_{A \times A'}$ is defined by $\{((q,q'),(p,p')) \mid q \neq_A p \text{ or } q \neq_{A'} p\}$. □

Prop 2 also holds for the class of languages defined by positive or negative TAGEDs. A TAGED is *deterministic* if all rules have different left-hand sides

(hence there is at most one run per tree). For a deterministic TAGED A, we can prove that one can compute a non-deterministic TAGED accepting the complement of $L(A)$: 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 3. *TAGEDs are not determinizable.*

Proof. Let $\Sigma = \{f, a\}$ an alphabet where f is binary and a constant. Consider the language $L_0 = \{f(t,t) \mid t \in T_\Sigma\}$ of Example 2. It is obvious that L_0 is definable by a non-deterministic (bounded) TAGED. Suppose that there is a deterministic TAGED $A = (\Sigma, Q, F, \Delta, =_A, \neq_A)$ such that $L(A) = L_0$. Let t be a tree whose height is strictly greater than $|Q|$. Since $f(t,t) \in L_0$, there are a successful run $q_f(r,r)$ of A on $f(t,t)$ for some final state q_f, two nodes u, v and a state $q \in Q$ such that $t|_v$ is a strict subtree of $t|_u$, and $O_r(u) = O_r(v) = q$. Since $f(t|_u, t|_u) \in L_0$, and A is deterministic, there is a final state $q'_f \in F$ and a rule $f(q,q) \to q'_f \in \Delta$. Hence $q'_f(r|_u, r|_v)$ is a run of A on $f(t|_u, t|_v)$. Since $q_f(r,r)$ satisfies the constraints, $q'_f(r|_u, r|_v)$ also satisfies the constraints. Hence $f(t|_u, t|_v) \in L(A)$, which contradicts $t|_u \neq t|_v$. □

Proposition 3 is not surprising, since:

Proposition 4. *The class of TAGED-definable languages is not closed by complement.*

Proof. (Sketch) We exhibit a tree language whose complement is easily definable by a TAGED, but which is not TAGED-definable. This language is the union of sets T_n, for all $n \in \mathbb{N}$, where $T_n = \{f(g(t,t), t') \mid t \in T_\Sigma, t' \in T_{n-1}\}$, and $T_0 = \{a\}$. To check whether a tree is in T_n, a TAGED would have to perform n equality tests, for each subtree rooted by g. This would require n states. This is only an intuition. The proof is a bit more complicated as the TAGED could also perform inequality tests. □

We end up this section with an undecidability result:

Proposition 5. *Testing universality of TAGEDs is undecidable.*

Proof. (Sketch) We adapt the proof of [18] for undecidability of emptiness of classical tree automata with equality constraints. We start from an instance of the Post Correspondence Problem (PCP). We encode the set of solutions of PCP as a tree language whose complement is easily definable by a TAGED. Hence, the complement is universal iff PCP has no solution. □

Even if TAGEDs are not determinizable, we can assume that testing an equality between subtrees can be done using the same state, as stated by the following lemma:

Lemma 1. *Every TAGED A is equivalent to a TAGED A' (whose size might be exponential in the size of A) such that $=_{A'} \subseteq id_{Q_{A'}}$, where $id_{Q_{A'}}$ is the identity relation on $Q_{A'}$. Moreover, A' can be built in exponential time (and may have exponential size).*

Proof. (Sketch) 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 equal by A in the run r (i.e. $\forall u, v \in U,\ O_r(u) =_A O_r(v)$). 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[1], since only subtrees which have been successfully tested to be equal are shared. We construct A' such that it simulates a run on this DAG, obtained by overlapping the runs on every equal subtrees for which a test has been done. □

4 Emptiness of Positive and Negative TAGEDs

In this section we prove decidability of emptiness of positive and negative TAGEDs respectively. For positive TAGEDs, it uses Lemma 1, and the classical reachability method for tree automata. For negative TAGEDs, we reduce the problem to testing satisfiability of set constraints.

Theorem 1. *Testing emptiness of positive TAGEDs is EXPTIME-complete.*

Proof. **upper bound** Let A be a positive TAGED such that its equality relation is a subset of the identity relation (otherwise we transform A modulo an exponential blow-up, thanks to Lemma 1). Let A^- be its associated tree automaton (i.e. A without the constraints). We have $L(A) \subseteq L(A^-)$.
Then it suffices to apply a slightly modified version of the classical reachability method used to test emptiness of a tree automaton [7]. In particular, we can make this procedure associate with any state q a unique tree t_q. When a new state is reached, it can possibly activate many rules $f(q_1, q_2) \to q$ whose rhs are the same state q. The algorithm has to make a choice between this rules in order to associate a unique tree $t_q = f(t_{q_1}, t_{q_2})$ to q. This choice can be done for instance by giving an identifier to each rule and choosing the rule with the least identifier.
If $L(A^-)$ is empty, then $L(A)$ is also empty. If $L(A^-)$ is non-empty, we get a tree t and a run r which obviously satisfies the equality constraints, since a state q such that $q =_A q$ is mapped to unique tree t_q (if q is reachable).
lower bound We reduce the problem of testing emptiness of the intersection of n tree automata $A_1, \ldots, A_n$ (see [7]), which is known to be EXPTIME-complete. We assume that their sets of states are pairwise disjoint ($Q_i \cap Q_j = \varnothing$ whenever $i \neq j$), and for all $i = 1, \ldots, n$, A_i has exactly one final state q_{f_i}, and q_{f_i} does not occur in lhs of rules of A_i (otherwise we slightly modify A_i, modulo a factor 2 in the size of A_i). We let $L = \{f(t_1, \ldots, t_n) \mid f \in \Sigma, \forall i,\ t_i \in L(A_i), \forall i, j,\ t_i = t_j\}$. It is clear that L is empty iff $L(A_1) \cap \ldots L(A_n)$ is empty. It is not difficult to construct a TAGED A (with $|A| = O(\sum_i |A_i|)$), such that $L = L(A)$: it suffices to take the union of $A_1, \ldots, A_n$ and to add the rule $f(q_{f_1}, \ldots, q_{f_n}) \to q_f$, where q_f is a fresh final state of A. Then we add the following equality constraints: $\forall i, j,\ q_{f_i} =_A q_{f_j}$. □

[1] there might be two isomorphic subgraphs occurring at different positions.

If $=_A \subseteq id_Q$, in a successful run we can assume that the subruns rooted at states q such that $q =_A q$ are the same. Hence, we can introduce a pumping technique for positive TAGEDs satisfying this property. The idea is to pump similarly in parallel below all states q such that $q =_A q$, while keeping the equality constraints satisfied. The pumping technique is described in the full version of the paper [24]. Thanks to this, if there is a loop in a successful run, we can construct infinitely many accepted trees. In particular:

Theorem 2. *Let A be a positive TAGED. It is decidable whether $L(A)$ is infinite or not, in $O(|A||Q|^2)$ if $=_A \subseteq id_Q$, and in EXPTIME otherwise.*

We now prove decidability of emptiness of negative TAGEDs ($=_A = \varnothing$), by reduction to positive and negative set constraints (PNSC for short). Set expressions are built from set variables, function symbols, and Boolean operations. 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 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 [5, 1, 22, 12]. In particular, it is known to be NEXPTIME-complete. We do not formally define set constraints and refer the reader to [5, 1, 22, 12].
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. 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 all state $q \in Q$ to occur in the rhs of a rule. We associate with A the system S_A defined by:

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

We can prove that $L(A)$ is non-empty iff S_A has a solution. Let $(A, \neq_A)$ be a negative 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 $q \neq_A p$. We can prove that $L(A, \neq_A) \neq \varnothing$ iff S'_A has a solution. Since deciding existence of a solution of a system of PNSC is in NEXPTIME, we get:

Theorem 3. *Emptiness of negative TAGEDs is decidable in NEXPTIME.*

5 Emptiness when Mixing Equality and Disequality Constraints

In this section, we mix equality and disequality constraints. This has already been done in [10] for bounded TAGEDs. Emptiness was proved by decomposition of runs, but here we use a pumping technique that allows to decide emptiness for a class of TAGEDs that significantly extends the class considered in [10]. In particular, we allow an unbounded number of positive tests, but boundedly

many negative tests along root-to-leaves paths, *i.e.* branches. While this class subsumes positive TAGEDs, the upper-bound for testing emptiness is bigger than the bound obtained in Section 4.
Formally, a *vertically bounded TAGED* (vbTAGED for short) is a pair (A, k) where A is a TAGED, 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 $\mathrm{dom}(\neq_A)$ occurring along a root-to-leaves path is bounded by k: in other words, for all root-to-leaves path $u_1 \lhd \ldots \lhd u_n$ of t (where each u_i is a node), one has $|\{u_i \mid O_r(u_i) \in \mathrm{dom}(\neq_A)\}| \leq k$.
We now come to the main result of the paper:

Theorem 4. *Emptiness of vbTAGEDs (A, k) is decidable in 2NEXPTIME.*

Proof. Sketch We first transform A so that it satisfies $=_A \subseteq id_Q$, thanks to Lemma 1 (modulo an exponential blow-up). Let $t \in T_\Sigma$, and r a run of A on it which satisfies the equality constraints (but not necesssarily the disequality constraints), and such that its root is labeled by a final state. We introduce sufficient conditions on t and r (which can be verified in polynomial-time, in $|t|$, $|r|$ and $|A|$) to be able to repair the unsatisfied inequality constraints in t in finitely many rewriting steps. These rewritings can be done while keeping the equality constraints satisfied. In particular, since $=_A \subseteq id_Q$, we can assume that for all $u, v \in N_t$ such that $u \sim_{t,r} v$, $r|_u = r|_v$. Hence, we can use a "parallel" pumping technique similar to the pumping technique for positive TAGEDs. The pumping is a bit different however: indeed, if t and r satisfies the sufficient conditions, we increase the size of some contexts of t and r, called *elementary contexts*, in order to repair all the unsatisfied inequality constraints. The repairing process is inductive. In particular, we introduce a notion of frontier below which all inequality constraints have been repaired. The process stops when the frontier reach the top of the tree (and in this case the repaired tree is in the language). From a tree and a run that satisfy the sufficient conditions, and a frontier F, one can create a new tree and a new run satisfying the conditions, and a new frontier which is strictly contained in F. Conversely, if $L(A, k) \neq \varnothing$, then there is a tree t and a run r satisfying the conditions such that the height of t is smaller than $2(k + |Q|)|Q|$ (and by $(k + 2^{|Q|})2^{|Q|+1}$ if $=_A \not\subseteq id_Q$). Hence, it suffices to guess a tree and a run satisfying the conditions to decide emptiness of A.

Since the class of vbTAGEDs subsumes the class of positive TAGEDs, we also get an EXPTIME lower bound for emptiness of vbTAGEDs, by Theorem 1. Moreover, if $=_A \subseteq id_Q$ and $k \leq |Q|$ (or k is unary encoded), emptiness of A is in NEXPTIME. □

6 Applications

6.1 MSO with Tree Isomorphism Tests

We study an extension of MSO with isomorphism tests between trees. Trees over an alphabet Σ are viewed as structures over the signature consisting of unary

predicates O_a, for all $a \in \Sigma$, to test the labels, and the two successor relations S_1 and S_2 which relates the parent to its first child and its second child respectively. The domain of the structure is the set of nodes.
We consider node variables x, y and set variables X, Y. MSO consists of the closure of atomic formulas $O_a(x)$ (for $a \in \Sigma$), $S_1(x, y)$, $S_2(x, y)$, $x \in X$, by conjunction $\wedge$, negation $\neg$, and existential quantifications $\exists x$, $\exists X$. We refer the reader to [17] for the semantics of MSO. It is well-known that MSO sentences and tree automata define the same tree languages [23]. We use similar back and forth translations to prove that an extension of MSO with tree isomorphism tests effectively defines the same language as vertically bounded TAGED. This significantly improves the result of [10].
We consider a predicate $\mathsf{eq}(X)$, which holds in a tree t under assignment $\rho : X \mapsto U$ (denoted by $t, \rho \models \mathsf{eq}(X)$), for some $U \subseteq N_t$, if for all $u, v \in U$, the trees $t|_u$ and $t|_v$ are isomorphic. For all $k \in \mathbb{N}$, we consider the predicate $\mathsf{diff}_k(X, Y)$, which holds in t under assignment ρ if (i) the maximal length of a descendant chain in $\rho(X)$ and $\rho(Y)$ is bounded by k, (ii) for all $u \in \rho(X), v \in \rho(Y)$, the trees $t|_u$ and $t|_v$ are **not** isomorphic. We consider $\mathrm{MSO}^{\exists}_{=}$ the extension of MSO whose formulas are of the form $\exists X_1 \ldots \exists X_n \phi$, where ϕ is an MSO formula extended with atoms $\mathsf{eq}(X_i)$ and $\mathsf{diff}_k(X_i, X_j)$ $(1 \leq i, j \leq n)$[2]. $\mathrm{MSO}^{\exists}_{=}$ is strictly more expressive than MSO as tree isomorphism is not expressible in MSO [7], but as a corollary of Theorem 4, we obtain:

Theorem 5. *$\mathrm{MSO}^{\exists}_{=}$ and vbTAGEDs effectively define the same tree languages, and satisfiability of $\mathrm{MSO}^{\exists}_{=}$ formulas is decidable.*

If we allow universal quantification of set variables $X_1, \ldots, X_n$, the logic becomes undecidable (even if the X_is denote singletons) [10].

6.2 Unification with Membership Constraints

We show that TAGEDs are particularly suitable to represent sets of ground instances of terms. Then we investigate a particular unification problem with tree and context variables where context variables can occur only in a restricted manner. In particular, we consider first-order logic (FO) over term equations $t \approx t'$, where t, t' are terms with tree and context variables, such that in a formula, every context variable can occur at most once. Tree and context variables might be constrained to range over regular languages (membership constraints). We prove this logic to be undecidable and exhibit a decidable existential fragment. This is particularly relevant for XML queries, as we can express tree patterns with negations. For instance, let L_{dtd} be a regular tree language representing the DTD of a bibliography, d a ground term representing a bibliography, L_{path} the set of unary contexts denoted by the XPath expression *bib/books* (i.e. contexts whose hole is reachable by the path *bib/books*), and X a unary context variable. The formula $\phi(y, z) = \exists X, d \approx X(book(author(y), title(z))) \wedge d \in L_{dtd} \wedge X \in L_{path}$ checks that d conforms to the DTD and extracts from d all (author,title) pairs reachable from the root by a path *bib/books/book*. The formula $\exists z \exists z',\ \phi(y, z) \wedge$

[2] We assume that $X_1, \ldots, X_n$ are not quantified in ϕ

$\phi(y, z') \wedge \neg(z \approx z')$ extracts from d all authors y who published at least two books.
The restriction on context variables allows to test (dis)equalities arbitrarily deeply but can not be used to test context (dis)equalities. Even with this restriction, FO is undecidable, while it is known that without context variables, FO on atoms $t \approx t'$ with membership constraints is decidable [8].
Let Σ be a ranked alphabet (assumed to be of binary and constant symbols for the sake of clarity). Let $\mathcal{X}_t$ be a countable set of tree variables x, y, and $\mathcal{X}_c$ a countable set of multi-ary context variables X, Y (we assume the existence of a mapping $\mathtt{ar} : \mathcal{X}_c \to \mathbb{N}$ giving the arity of any context variable). The set of terms over Σ, $\mathcal{X}_t$ and $\mathcal{X}_c$ is denoted by $\mathcal{T}(\Sigma, \mathcal{X}_t, \mathcal{X}_c)$. For instance $X(f(x, X(y), x))$ is a term where $X \in \mathcal{X}_c$ (arity 1), $f \in \Sigma$ (arity 3) and $x, y \in \mathcal{X}_t$. A term is *ground* if it does not contain variables. The set of ground terms over Σ is simply denoted T_Σ. We also denote by $\mathcal{C}_\Sigma$ the set of contexts over Σ, and by $\mathcal{C}_\Sigma^n$ the set of n-ary contexts over Σ, for all $n \in \mathbb{N}$. For all $C \in \mathcal{C}_\Sigma^n$, and terms $t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{X}_t, \mathcal{X}_c)$, we denote by $C[t_1, \ldots, t_n]$ the term obtained by substituting the holes in C by $t_1, \ldots, t_n$ respectively (see [7] for a formal definition of contexts).
A *ground substitution* σ is a function from $\mathcal{X}_t \cup \mathcal{X}_c$ into $\mathcal{T}_\Sigma \cup \mathcal{C}_\Sigma$ such that for all $x \in \mathcal{X}_t$, $\sigma(x) \in T_\Sigma$, and for all $X \in \mathcal{X}_c$, $\sigma(X) \in C_\Sigma$ and $\mathtt{ar}(X) = \mathtt{ar}(C)$. The ground term obtained by applying σ on a term t is denoted $t\sigma$. A ground term t' is a *ground instance* of a term t if there is σ such that $t' = t\sigma$. Finally, a term t is *context-linear* if every context variables occurs at most once in t.

Proposition 6. *Let $t \in \mathcal{T}(\Sigma, \mathcal{X}_t, \mathcal{X}_c)$ be context-linear. The set of ground instances of t is definable by a positive TAGED.*

Proof. (Sketch) It suffices to introduce states for each subterm of t, and a special state $q_\forall$ in which every ground term evaluates. Then we add state equalities $q_x =_A q_x$ for all variable x occurring in t. □

We now introduce unification problems. An *equation* e is a pair of terms denoted by $t \approx t'$, where $t, t' \in \mathcal{T}(\Sigma, \mathcal{X}_t, \mathcal{X}_c)$. A ground substitution σ is a solution of e if $t\sigma$ and $t'\sigma$ are ground terms, and $t\sigma = t'\sigma$. Let $n \in \mathbb{N}$. A regular n-ary context language L is a regular language over $\Sigma \cup \{\circ_1, \ldots, \circ_n\}$, where $\circ_1, \ldots, \circ_n$ are fresh symbols denoting the holes, and such that every symbol $\circ_i$ occurs exactly once in terms (this can be ensured by a regular control). A *membership constraint* is an atom of the form $x \in L_x$, or $X \in L_X$, where $x \in \mathcal{X}_t$, $X \in \mathcal{X}_c$, L_x is a regular tree language, and L_X is a regular $\mathtt{ar}(X)$-ary context language.
We consider FO over equations and membership constraint atoms, with the following restriction: for all formulas ϕ, and all context variables $X \in \mathcal{X}_c$, there is at most one equation e, and one term t in e such that X occurs in t. We denote by FO$[\approx, \in]$ this logic. FO$[\approx, \in]$-formulas are interpreted over ground substitutions σ. We define the semantics $\sigma \models \phi$ inductively: $\sigma \models e$ if σ is a solution of e, $\sigma \models x \in L_x$ if $\sigma(x) \in L_x$ (and similarly for $X \in L_X$), $\sigma \models \exists x\phi$ if there is a ground term t such that $\sigma[x \mapsto t] \models \phi$ (and similarly for $\exists X\phi$). Disjunction and negation are interpreted as usual.
We can show the following by reducing PCP:

Proposition 7. *Satisfiability of FO*$[\approx, \in]$ *is undecidable.*

However, it is known that satisfiability of FO$[\approx, \in]$ in which no context variable occurs is decidable [8]. We consider the existential fragment FO$^{\exists}[\approx, \in]$ of FO$[\approx, \in]$ formulas where existential quantifiers $\exists x$ or $\exists X$ cannot occur below an odd number of negations.

Theorem 6. *Satisfiability of FO*$^{\exists}[\approx, \in]$ *is decidable.*

Proof. (Sketch) Wlog, we consider only closed formulas. We define a normal form which intuitively can be viewed as a set of pairs (E, M), where E is a set of equations e (or negated equations $\neg e$), and M is a set of membership constraints. For each pair (E, M), we construct a vbTAGED $(A_{E,M}, |E|)$ which defines the ground instances of the term t_0 depicted in Fig. 1 satisfying: (i) # is a fresh symbol, (ii) for all terms t, t', there exists $i \in \{1, \ldots, n\}$ s.t. $t = t_i$ and $t' = t'_i$ iff either $(t \approx t') \in E$ or $\neg(t \approx t') \in E$, (iii) if $t_0\sigma$ is a ground instance of t_0, then the membership constraints are satisfied, and σ is a solution of every equation of E (this can be done for instance by adding state inequalities $q_t \neq_A q_{t'}$, if $\neg(t \approx t') \in E$). The formula is satisfiable iff there is a pair (E, M) such that $L(A_{E,M}, |E|) \neq \varnothing$. □

Anti-pattern matching [15] considers terms with negations (called anti-patterns). For instance, the anti-pattern $f(x, \neg x)$ denotes all the ground terms $f(t_1, t_2)$ such that $t_1 \neq t_2$. More generally, negations can occur at any position in the term: $\neg(g(\neg a))$ denotes all ground terms which are not rooted by g or $g(a)$, and $\neg f(x, x)$ denotes ground terms which are not of the form $f(t, t)$. A ground term matches an anti-pattern if it belongs to its denotation. [15] proves it to be decidable. We can easily define a vbTAGED A_p which accepts the denotation of an anti-pattern p where negations occur at variables only. Thus the anti-pattern matching problem reduces to test membership to $L(A_p)$. When negations occur arbitrarily, the translation is not so clear since the semantics of anti-patterns is universal (a ground term t matches $\neg f(x, x)$ if $\forall x, t \neq f(x, x)$). We let as future work this translation (for instance by pushing down the negations).

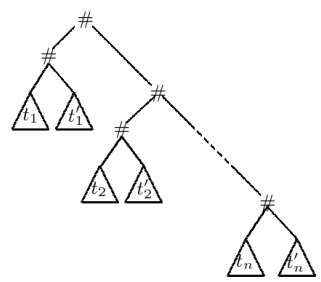

Fig. 1. term t_0

7 Future Work

In [10], TAGEDs are based on hedge automata [19], so that they accept unranked trees. We can encode unranked trees over Σ as terms over the signature $\Sigma \cup \{\mathsf{cons}\}$, where cons is a binary symbol denoting concatenation of an unranked tree to an hedge. For instance, $f(a, b, c)$ maps to $f(\mathsf{cons}(a, \mathsf{cons}(b, c)))$. Hedge automata can be translated into tree automata over those encodings.

Moreover, the encoding is unique, and any subtree t becomes a subtree rooted by a symbol of Σ in the encoding. Hence testing constraints between subtrees in unranked trees is equivalent to test constraints between subtrees rooted by Σ in binary encodings. Therefore, TAGEDs over unranked trees can be translated into TAGEDs over encodings, and we can prove that all the results presented in this paper carry over to unranked trees. Moreover, in [10], we consider the TQL logic over unranked trees, and prove a fragment of it to be decidable, by reduction to emptiness of bounded TAGEDs (over unranked trees). This work could be used to decide larger fragments of TQL, via a binary encoding. Concerning the unification problem considered here, we would like to use TAGEDs to test whether there are finitely many solutions, and to represent the set of solutions. A question remains: deciding emptiness of full TAGEDs. It is not easy, even for languages of trees of the form $f(t_1, t_2)$, where t_1 and t_2 are unary. Finally, it could be interesting to consider more general tests, like recognizable relations on trees (since tree (dis)equality is a particular recognizable binary relation).

References

1. Alexander Aiken, Dexter Kozen, and Edward L. Wimmers. Decidability of systems of set constraints with negative constraints. *Information and Computation*, 122(1):30–44, 1995.
2. Siva Anantharaman, Paliath Narendran, and Michael Rusinowitch. Closure properties and decision problems of dag automata. *Inf. Process. Lett.*, 94(5):231–240, 2005.
3. B. Bogaert and S. Tison. Equality and disequality constraints on direct subterms in tree automata. In *STACS'92*, volume 577 of *LNCS*, pages 161–171.
4. Witold Charatonik. Automata on dag representations of finite trees. Technical report, 1999.
5. Witold Charatonik and Leszek Pacholski. Set constraints with projections are in NEXPTIME. In *IEEE Symposium on Foundations of Computer Science*, 1994.
6. H. Comon and V. Cortier. Tree automata with one memory, set constraints and cryptographic protocols. *TCS*, 331(1):143–214, 2005.
7. H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available at http://www.grappa.univ-lille3.fr/tata, 2007.
8. Hubert Comon and Catherine Delor. Equational formulae with membership constraints. *Information and Computation*, 112(2):167–216, 1994.
9. M. Dauchet, A.-C. Caron, and J.-L. Coquidé. Reduction properties and automata with constraints. *JSC*, 20:215–233, 1995.
10. E. Filiot, J-M. Talbot, and S. Tison. Satisfiability of a spatial logic with tree variables. In *CSL'07*, pages 130–145.
11. E. Filiot. *Logics for n-ary queries in trees.* PhD thesis, Lille University, 2008.
12. Rémi Gilleron, Sophie Tison, and Marc Tommasi. Some new decidability results on positive and negative set constraints. In *Proceedings of the International Conference on Constraints in Computational Logics*, pages 336–351, 1994.
13. F. Jacquemard, M. Rusinowitch, and L. Vigneron. Tree automata with equality constraints modulo equational theories. Research Report, LSV, ENS Cachan, 2006.

14. W. Karianto and C. Löding. Unranked tree automata with sibling equalities and disequalities. Research Report, RWTH Aachen,2006.
15. Claude Kirchner, Radu Kopetz, and Pierre-Etienne Moreau. Anti-pattern matching. In *ESOP'07*, 2007.
16. Temur Kutsia and Mircea Marin. Solving regular constraints for hedges and contexts. In *UNIF'06*, pages 89–107.
17. L. Libkin. Logics over unranked trees: an overview. *LMCS'06*, 3(2):1–31, 2006.
18. J. Mongy. *Transformation de noyaux reconnaissables d'arbres. Forêts RATEG.* PhD thesis, Université de Lille, 1981.
19. M. Murata. Hedge automata: A formal model for xml schemata. Technical report, Fuji Xerox Information Systems, 1999.
20. Frank Neven. Automata, logic, and xml. In *CSL'02*, pages 2–26.
21. T. Schwentick. Automata for xml – a survey. *J. Comput. Syst. Sci. 73*, 3 (2007), 289–315.
22. Kjartan Stefansson. Systems of set constraints with negative constraints are nexptime-complete. In *LICS*, 1994.
23. J. W. Thatcher and J. B. Wright. Generalized finite automata with an application to a decision problem of second-order logic. *Mathematical System Theory*, 2:57–82, 1968.
24. Full paper version. Available at http://hal.inria.fr/inria-00292027.

A Closure Properties of TAGEDS and Decision Problems

A.1 Proof of Proposition 2

It remains to prove closure by intersection.

Proof. Let $A = (\Sigma, Q, F, \Delta, =_A, \neq_A)$ and $A' = (\Sigma, Q', F', \Delta', =_{A'}, \neq_{A'})$ be two TAGED. Let $\text{ta}(A) = (\Sigma, Q, F, \Delta)$ and $\text{ta}(A') = (\Sigma, Q', F', \Delta')$ be their respective tree automaton parts.
We let $\text{ta}(A) \times \text{ta}(A')$ be the product automaton of $\text{ta}(A)$ and $\text{ta}(A')$, whose set of final states is $F \times F'$ (see [7] for more details about product construction). We let $A \times A'$ be defined by $(\text{ta}(A) \times \text{ta}(A'), =_{A\times A'}, \neq_{A\times A'})$ where:

- $=_{A\times A'} = \{((q, q'), (p, p') \mid q =_A p \text{ or } q =_{A'} p\}$
- $\neq_{A\times A'} = \{((q, q'), (p, p') \mid q \neq_A p \text{ or } q \neq_{A'} p\}$

Now we prove that $L(A \times A') = L(A) \cap L(A')$.
Let $t \in L(A \times A')$, then there exists a successful run r'' of $A \times A'$ on t. By construction, this run can be decomposed into a successful run r of $\text{ta}(A)$ on t and a successful run r' of $\text{ta}(A')$ on t. It remains to show that r and r' satisfies the constraints of A and A' respectively. Let $u, v \in N_t$ such that $O_r(u) =_A O_r(v)$. It means that there exists four states $q, p \in Q, q', p' \in Q'$ such that $O_{r''}(u) = (q, q')$, $O_{r''}(v) = (p, p')$ and $q =_A p$. Hence $(q, q') =_{A\times A'} (p, p')$ and we get $t|_u = t|_v$. This is similar for r' and when considering inequalities.
Conversely, suppose that $t \in L(A) \cap L(A')$. Hence there exists two successful runs r and r' of A and A' on t respectively. By construction of $A \times A'$, these two runs can be combined into a run r'' of $A \times A'$ on t. It remains to show that r'' satisfies the constraints of $A \times A'$. Let $u, v \in N_t$ such that $O_{r''}(u) =_{A\times A'} O_{r''}(v)$. It means that there exists four states $q, p \in Q$ and $q', p' \in Q'$ such that $O_{r''}(u) = (q, q')$, $O_{r''}(v) = (p, p')$, and either $q =_A p$ or $q' =_{A'} p'$. If $q =_A p$, we easily get $t|_u = t|_v$ (since r respects the constraints of A), and similarly if $q' =_{A'} p'$. It similar when dealing with inequalities. □

A.2 Proof of Proposition 4

We let $\Sigma = \{f, g, a\}$ where f, g are binary and a is a constant, and $h \notin \Sigma$ be a binary symbol. We let $T_0 = \{a\}$, and for $n > 0$, $T_n = \{f(g(t, t), t') \mid t \in T_{\{h,a\}},\ t' \in T_{n-1}\}$. We let $L = \bigcup_{n\in\mathbb{N}} T_n$. The complement of L is easily definable by a TAGED. Suppose that L is definable by a TAGED $A = (\Sigma, Q, F, \Delta, =_A, \neq_A)$. 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 2). It is clear that $t_n \in T_n$. Hence there is a successful run r of A on t_n. 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 $u, u' \in N_{t_n}$, and a state $q \in Q$ such that: (i) $u = 2^{i_0}1b$ and $u' = 2^{j_0}1b'$, (ii) $O_r(u) = O_r(u') = q$. We let $t'_n = t_n[t_n|_u]_{u'}$, i.e. the tree t_n where the subtree at node u' has been substituted by the subtree at node u. We do the same corresponding substitution in r, which result in a run denoted r'. Note that $t'_n \notin L$ since $|\alpha_i| \neq |\alpha_j|$, for

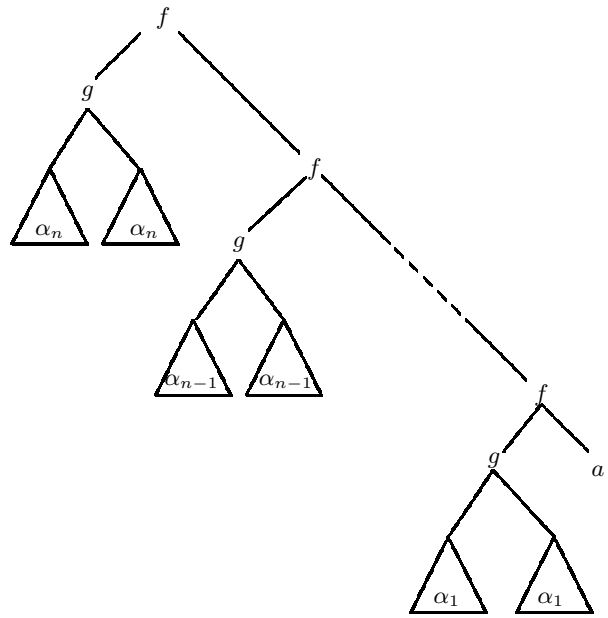

Fig. 2. Tree t_n

$i \neq j$, by definition of t_n, for any $i \neq j$. We now prove that r' is successful (i.e. satisfies the constraints), which would contradict $t_n|_u \neq t_n|_{u'}$. Let $N = \{2^k \mid k = 0, \ldots, j_0\} \cup \{2^{j_0}1\}$. Let $v, w \in N_{t'_n}$, $v \neq w$. We consider three cases (the other are symmetric):

- if $v, w \notin N$, then necessarily the constraints are still satisfied.
- suppose that $v \in N$ is of the form 2^k, for $k \leq n-1$. We prove that necessarily, $t'_n|_v \neq t'_n|_w$. Indeed, the root of $t'_n|_v$ is necessarily labeled f. If the root of $t'_n|_w$ is labeled f, then either $t'_n|_v$ is a strict subtree of $t'_n|_w$ or conversely. Otherwise the root of $t'_n|_w$ is labeled by g, h or a, so we obviously have $t'_n|_v \neq t'_n|_w$.
  Hence, if $O_{r'}(v) \neq_A O_{r'}(w)$, the constraints is satisfied. We can easily prove that $O_{r'}(v) =_A O_{r'}(w)$ is impossible.
- suppose that $v \in N$ is of the form $2^{j_0}1$. The root of $t'_n|_v$ is necessarily labeled g. There are two cases:
  - if $O_{r'}(v) =_A O_{r'}(w)$, we can prove a contradiction.
    Indeed, if w is of the form $2^{j_0}1b'w'$, for some $w' \in N_{\alpha_{j_0}}$, we have $O_r(v) =_A O_r(2^{i_0}1b'w')$, and $t_n|_{2^{i_0}1b'w'} = t_n|_v$, which is impossible since the root of $t_n|_{2^{i_0}1bw'}$ is labeled h or a, and the root of $t_n|_v$ is labeled g.
    If w is of the form 2^k, for some $k \in \{0, \ldots, n\}$, then we have $t_n|_w = t_n|_v$, which is also impossible for similar reasons.
    If $w = 2^k1$, for some $k \in \{0, \ldots, n-1\}$, then $k \neq j_0$, and necessarily, we would have $t_n|_w = t_n|_v$, which is impossible since in this case we

would have $t_n|_w = g(\alpha_k, \alpha_k)$ and $t_n|_v = g(\alpha_{j_0}, \alpha_{j_0})$, which contradicts $|\alpha_k| \neq |\alpha_{j_0}|$.
Finally, if w is of the form $2^j 1cw'$, for some $j \neq j_0$, $c \in \{1, 2\}$, and $w' \in N_{\alpha_j}$. Hence we have $O_r(v) =_A O_r(w)$, and $t_n|_v = t_n|_w$, which contradicts that their respective roots are labeled with different labels.

- if $O_r(v) \neq_A O_r(w)$, then the constraint is satisfied, i.e. $t'_n|_v \neq t'_n|_w$. Suppose that $t'_n|_v = t'_n|_w$, since the root of $t'_n|_v$ is labeled g, w is necessarily of the form $2^k 1$, with $k \neq j_0$. Hence $t_n|_w$ is of the form $g(\alpha_{i_0}, \alpha_{j_0}) = t'_n|_v$, which contradicts $t_n \in L$, since $|\alpha_{i_0}| \neq |\alpha_{j_0}|$.

Hence r' is a successful run of A on t'_n, which contradicts $t'_n \notin L$. □

A.3 Proof of Proposition 5

We reduce the Post Correspondence Problem (PCP)[3].
We adapt the proof of emptiness undecidability of automata with equality constraints of [18]. In this proof, it is shown that one can construct an automata with equality constraints which can recognize the solution of an instance of the Post Correspondence Problem (PCP).
Let Σ be an alphabet, and $\mathcal{I} = \{u_1, \ldots, u_m, v_1, \ldots, v_m\}$ be an instance of PCP, $\forall i \in \{1, \ldots, m\}, u_i, v_i \in \Sigma^*$. We denote by $\Sigma \cup \{f, c\}$ the ranked alphabet obtained by extending Σ with a fresh ternary function symbol f and a constant 0. Symbols from Σ are viewed as unary function symbols[4]
Let $(u_{i_1}, v_{i_1}), \ldots, (u_{i_n}, v_{i_n})$ be solution of $\mathcal{I}$. This solution can be represented as a term over $\Sigma \cup \{f, c\}$, as in Figure 3. For all $1 \leq j \leq m$, the notation $u_j(x)$ stands for the context $u_{j,1}(u_{j,2}(\ldots u_{j,k}(x))\ldots)$, where $u_{j,1}, \ldots, u_{j,k}$ are symbols from Σ and $u_j = u_{j,1} \ldots u_{j,k}$.
We let S be set of encodings of candidate solutions of the instance of PCP, i.e. S is the set of trees defined by the following grammar:

$$t \ ::= \ f(t_u, t, t_v) \mid c \quad t_u \ ::= \ u_i(t_u) \mid c \quad t_v \ ::= \ v_j(t_v) \mid c \quad j = 1, \ldots, m$$

The set S can easily be defined by a tree automaton. We can define a (bounded) TAGED A which checks whether an encoding of a candidate solution is non-valid (i.e. is not a solution of $\mathcal{I}$). In other words, $L(A) \cap S$ is the set of encodings of non-valid solutions. A needs to check if the input tree matches the pattern $f(X, _, \neg X)$, where X is a pattern variable which intuitively is matched against a whole subtree, $_$ matches everything, and $\neg X$ is intended to match the third subtree if it is different from the tree matched by X. In addition, A checks that one of the following patterns is matched against a subtree of the input tree:

$$f(u_i(_), _, v_j(_)), \ \forall i \neq j \qquad f(u_i(X), f(\neg X, _, _), _) \quad \forall i \in \{1, \ldots, m\}$$
$$f(_, f(_, _, \neg X), v_j(X)) \quad \forall j \in \{1, \ldots, m\}$$

[3] Remind that an instance of PCP is given by a finite alphabet Σ and $2m$ words $u_1, \ldots, u_m, v_1, \ldots, v_m$ over Σ. A solution to this problem is a finite sequence of indices $i_1, \ldots, i_n$ such that $u_{i_1} \ldots u_{i_n} = v_{i_1} \ldots v_{i_n}$.

[4] to simplify, we do not consider binary symbols, but the proof can easily be adapted to a binary signature

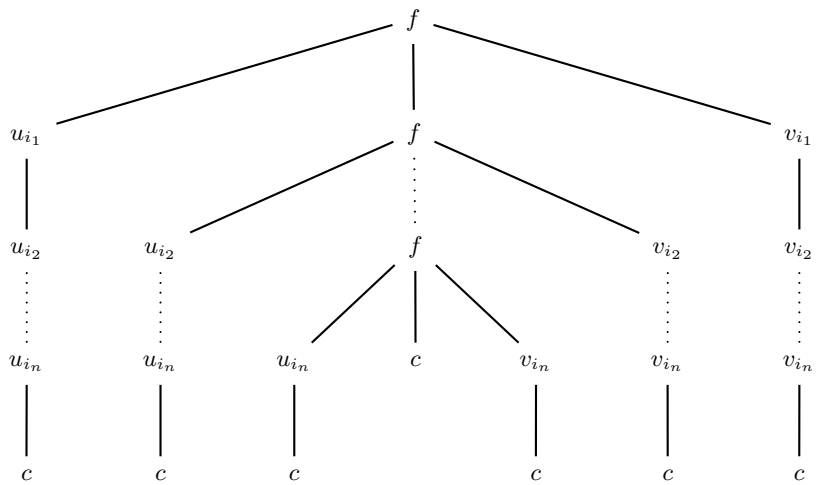

Fig. 3. representation of a solution of PCP

Hence, we have $\overline{L(A)} \cap S \neq \varnothing$ iff $\mathcal{I}$ has a (valid) solution, iff $T_\Sigma \not\subseteq L(A) \cup \overline{S}$. Hence, PCP reduces to (non) universality of (bounded) TAGED, since $L(A) \cup \overline{S}$ can be defined by a (bounded) TAGED, thanks to Proposition 2. □

A.4 Proof of Lemma 1

Let $A = (\Lambda, Q, F, \Delta, =_A, \neq_A)$. We start by defining useful notions, and then prove Lemma 1.

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

Node Equivalence Given a tree $t \in L(A)$ 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[5]. 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 two nodes u', v' above u, v resp. such that the downward path from u' to u is isomorphic to the downward path from v' to v and $O_r(u') =_A O_r(v')$.

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

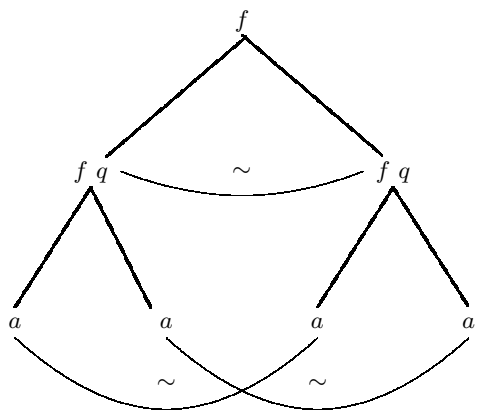

Fig. 4. Equivalence Relation where $q =_A q$

For instance, Fig 4 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 but all nodes are equivalent to themselves).
The following facts are implicitely used in the rest of the proof:

Proposition 8. *For all $u, v \in N_t$, if $u \sim_{t,r} v$, then $t|_u = t|_v$.*

Proof. If $u \leftrightarrow_{t,r} v$, there are u', v' such that $u' \trianglelefteq v'$ and $\text{path}_t(u', v')$ is isomorphic to $\text{path}_t(u, v)$, and $O_r(u') =_A O_r(v')$. Hence $t|_{u'} = t|_{v'}$ and therefore $t|_u = t|_v$. By transitivity, we also get $t|_u = t|_v$ whenever $u \sim_{t,r} v$. □

Proposition 9. *For all $u, v, u', v' \in N_t$ such that $u \lhd u'$ and $v \lhd v'$, if $u \sim_{t,r} v$ and the downward path from u to u' is isomorphic to the downward path from v to v', then $u' \sim_{t,r} v'$.*

Proof. This is true for $u \leftrightarrow_{t,r} v$, and by transitivity, it also holds for $u \sim_{t,r} v$.□

Proof of Lemma 1 We first describe the construction of the automaton and then prove its correctness.
Automaton Construction We define $Q' = 2^Q \times \mathcal{C}$ where $\mathcal{C}$ is a set of choices. Intuitively, when a choice has been made, it enforces the subtrees successfully tested to be equal to run in the same state.

For every $q \in \mathrm{dom}(=_A)$, we denote by $[q]$ its equivalence class by $=_A$. A *choice* is a 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 $\mathrm{dom}(c)$ may be strictly included into $\mathrm{dom}(=_A)$.
Let $c \in \mathcal{C}$. We let $Q_c = \{P \subseteq Q \mid \exists q \in P \cap \mathrm{dom}(=_A) \implies P = c(q)\}$ (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 let $=_c, \neq_c$ be defined by:

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

We let $A_c = (\Sigma, Q_c, F_c, \Delta_c, =_c, \neq_c)$, and let A' be the TAGED defining $\bigcup_{c \in \mathcal{C}} L(A_c)$ (we can construct it thanks to Proposition 2, and its size is exponential in the size of A). We now prove that $L(A') = L(A)$.
Correctness Let $q \in Q$. We let $L(q, A)$ be the set of trees t such that there exists a q-run[6] of A on t which satisfies the constraints. $L(q, A')$ is defined similarly.

$\boxed{L(A') \subseteq L(A)}$ Let $c \in \mathcal{C}$, $P \in Q_c$, and $t \in T_\Sigma$. such that $t \in L(P, A_c)$. 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. Moreover, if r_c satisfies the constraints, then r satisfies the constraints too. Indeed, let $u, v \in N_t$ such that $O_r(u) =_A O_r(v)$. By construction of r, we have $O_r(u) \in O_{r_c}(u)$ and $O_r(v) \in O_{r_c}(v)$. By definition of $=_c$, we get $t|_u = t|_v$. This is done similarly for inequalities.
$\boxed{L(A) \subseteq L(A')}$ Let $t \in L(A)$ and r a successful run of A on t. We let $\mathtt{states}(u) = \{q \mid \exists v \sim u, q = O_r(v)\}$. We let c be defined as follows: for all $p \in \mathrm{dom}(=_A)$, $u \in N_t$, $c(p) = \mathtt{states}(u)$ if $p \in \mathtt{states}(u)$. Hence, for every pair of equivalent nodes $u \sim v$, we have $c(O_r(u)) = c(O_r(v))$.
We show that there exists a run r_c of A_c on t. We define it by: $\forall u \in N_t$, $O_{r'}(u) = \mathtt{states}(u)$. Let us now show that r_c is a run of A_c, and that is satisfies the constraints.

- *for all $u \in N_t$, $\mathtt{states}(u) \in Q_c$.*
  By definition of $\mathtt{states}(u)$ and Q_c;
- *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 $f(\mathtt{states}(u_1), \mathtt{states}(u_2)) \to \mathtt{states}(u)$ is in Δ_c. Let $p \in \mathtt{states}(u)$. There is some node $v \in N_t$ such that $u \sim v$ and $O_r(v) = p$. Let $v1, v2$ be the two sons of v respectively (they exist since $t|_u = t|_v$). Let $p_1 = O_r(v_1)$ and $p_2 = O_r(v_2)$. 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 $p_1 \in \mathtt{states}(u_1)$ and $p_2 \in \mathtt{states}(u_2)$, which concludes the proof (this goes similarly for leaf nodes).

[6] a q-run is a run whose root is labeled by q

– 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 $O_{r_c}(u_1) = P_1$ and $O_{r_c}(u_2) = P_2$. 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, $c(p_1) = c(p_2) = P_1 = P_2 = \texttt{states}(u_1) = \texttt{states}(u_2)$.
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 $O_r(u'_1) = p_1$, $O_r(u'_2) = p_2$. 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 $t|_{u_1} = t|_{u_2}$.
– r_c *satisfies the disequality constraints*
It is similar to the previous case. Let $u_1, u_2 \in N_t$. If $O_{r_c}(u_1) \neq_c O_c(u_2)$, it means that there are two nodes $u'_1 \sim u_1$ and $u'_2 \sim u_2$ such that $O_r(u'_1) \neq_A O_r(u'_2)$. Since $t|_{u_1} = t|_{u'_1}$ and $t|_{u_2} = t|_{u'_2}$, and $t|_{u'_1} \neq t|_{u'_2}$, we get $t|_{u_1} \neq t|_{u_2}$.

□

B Negative TAGED

We sketch correctness of the system S_A. Suppose that S_A has a solution given by set of trees T_q, for all $q \in Q$. Let $q \in Q$ such that there is $t \in T_q$, $t \in L(A)$. We construct a run on t inductively.

– if $t = a \in \Sigma$, then we take the successful run reduced to the leaf q;
– if $t = f(t_1, t_2)$, for some $f \in \Sigma$, $t_1, t_2 \in T_\Sigma$. 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 $O_{r_1}(\epsilon) = q_1$ and $O_{r_2}(\epsilon) = q_2$. Hence $q(r_1, r_2)$ 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 $t \in L(A)$.
Conversely, if $L(A) \neq \varnothing$, there is a tree $t \in L(A)$ and a successful run r of A on t. For all $q \in Q$, we let $T_q = \{t|_u \mid u \in N_t, O_r(u) = q\}$. The set $\{T_q \mid q \in Q\}$ is a solution of S_A. □

C Pumping for TAGED: Proof of Theorems 2 and 4

In this section, we view trees t as particular graphs (and not terms), given by a pair (N_t, E_t^1, E_t^2), of vertices N_t, first-child edges E_t^1 and second-child edges E_t^2. In particular, N_t is an arbitrary set and not necessarily a prefix-closed set of words over $\{1, 2\}$. The notion of subtree carries over to the graph point of view of trees. Moreover, we confuse tree equality and tree isomorphism (hence, two equal trees have not necessarily the same set of nodes, but are isomorphic, i.e. their edge relations and label relations are preserved). We still denote by $\lhd$ the strict descendant relation and by $\unlhd$ its reflexive closure. Let $V \subseteq N_t$. We denote by $\uparrow_t V$ the set $\{w \mid \exists v \in V, w \unlhd v\}$. When it is clear from the context, we omit the subscript t.
We also confuse the nodes of t and the nodes of the runs of A on t, for some TAGED A. In other words, we always assume that $N_r = N_t$, for any run r on t.

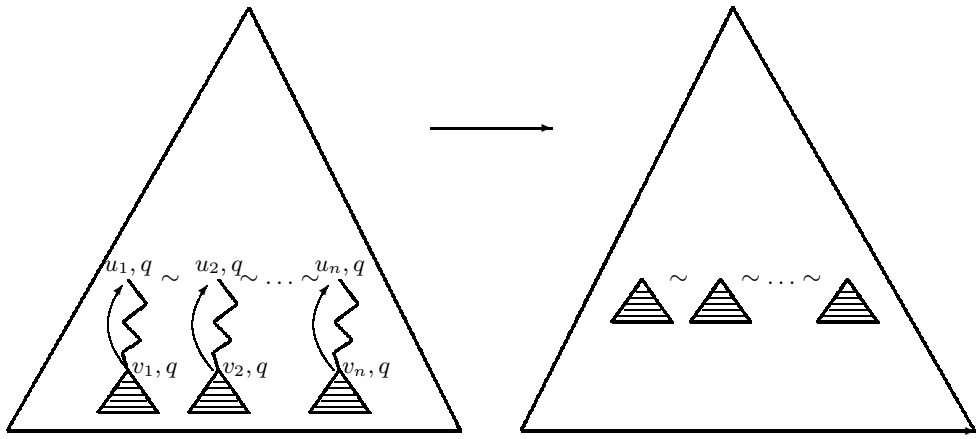

Fig. 5. Parallel pumping of state q

C.1 Parallel Pumping for Positive TAGED, and proof of Theorem 2

We first introduce a pumping technique for positive TAGEDs A which satisfy $=_A \subseteq id_Q$. The idea is to pump in parallel below states q such that $q =_A q$. Let $t \in T_\Sigma$ and r a successful run of A on t. Since $=_A \subseteq id_Q$, we can assume that we have the same subruns below states q such that $q =_A q$. Formally, for all $u, v \in N_t$ such that $O_r(u) =_A O_r(v)$, we assume that $r|_u = r|_v$ (otherwise we substitute $r|_u$ by $r|_v$, and the constraints are still satisfied). Suppose that there are two nodes $u, v \in N_t$ such that $u \lhd v$ and there is a state $q \in Q$ such that $O_r(u) = O_r(v) = q$. In general, we cannot pump the loop u, v because after pumping an equality constraint might be unsatisfied (in particular, if u is below a state from $\mathrm{dom}(=_A)$ which occurs twice in the run). Remind that $\leftrightarrow_{t,r}$ and $\sim_{t,r}$ are defined in Appendix A.4. Instead of pumping only below u, we pump in parallel below all nodes equivalent to u by $\sim_{t,r}$. This can be done since for all node u' such that $u \sim_{t,r} u'$, we have[7] $r|_u = r|_{u'}$. Now, let v' be the node below u' such that the downward path from u to v is isomorphic to the downward path from u' to v' (it exists since $t|_u = t|_{u'}$, by Prop 8). Since $r|_u = r|_{u'}$, we have $O_r(v') = O_r(u') = q$, so that we can also pump the loop u', v'. Hence, we can pump in parallel below all nodes equivalent to u while keeping the equality constraints satisfied.

Formally, the pumping technique is described by the following lemma:

[7] By definition of $\leftrightarrow_{t,r}$, for all nodes u_1, u_2 such that $u_1 \leftrightarrow_{t,r} u_2$, there are v_1, v_2 above u_1, u_2 respectively such that $\mathrm{path}_t(v_1, u_1)$ is isomorphic to $\mathrm{path}_t(v_2, u_2)$, and $O_r(v_1) =_A O_r(v_2)$. Since $=_A \subseteq id_Q$, we get $O_r(v_1) = O_r(v_2)$, and since we put the same run below states q such that $q =_A q$, we also get $r|_{v_1} = r|_{v_2}$, from which we deduce $r|_{u_1} = r|_{u_2}$. By transitivity, we also have $r|_{u_1} = r|_{u_2}$ for all nodes u_1, u_2 such that $u_1 \sim_{t,r} u_2$.

Lemma 2 (Pumping Lemma for Positive TAGED). *Let $t \in L(A)$, and r a successful run of A on t such that $\forall u, v \in N_t$, if $O_r(u) =_A O_r(v)$, then $r|_u = r|_v$. Let $\{u_1, \ldots, u_n\} \subseteq N_t$ be an $\sim_{t,r}$-equivalence class. Suppose that there is some state $q \in Q$ and some node v_1 such that $u_1 \lhd v_1$ and $O_r(u_1) = O_r(v_1) = q$. Let $v_2, \ldots, v_n$ such that for all $i \in \{2, \ldots, n\}$, $path_t(u_i, v_i)$ is isomorphic to $path_t(u_1, v_1)$. Let C_t (resp. C_r) be the n-ary context over Σ (resp. Q) such that $t = C_t[t|_{u_1}, \ldots, t|_{u_n}]$ (resp. $r = C_r[r|_{u_1}, \ldots, r|_{u_n}]$).*
Then $C_r[r|_{v_1}, \ldots, r|_{v_n}]$ is a successful run of A on $C_t[t|_{v_1}, \ldots, t|_{v_n}]$.

Proof. Fig. 5 illustrates this pumping. Let $r' = C_r[r|_{v_1}, \ldots, r|_{v_n}]$ and $t' = C_t[t|_{v_1}, \ldots, t|_{v_n}]$. It is clear that r' is a run on t' whose root is labeled by a final state. It remains to show that it satisfies the equality constraints. Let $u, v \in N_{t'}$ such that $O_{r'}(u) =_A O_{r'}(v)$. If neither v nor u is above one of the v_i, we have $t|_u = t'|_u$ and $t|_v = t'_v$, and $O_r(u) =_A O_r(v)$. Since r satisfies the equality constraints, we also have $t'|_u = t'|_v$.
Suppose now that $u \unlhd v_i$, for some $i \in \{1, \ldots, n\}$. By definition of $\sim_{t,r}$, we have $u \sim_{t,r} v$. We cannot have $u = v_i$, otherwise it would mean that q is an equality state, which is not possible since it would imply $t|_{u_1} = t|_{v_1}$, but $t|_{v_1}$ is a strict subtree of $t|_{u_1}$. Hence $u \lhd v_i$. It means that in t, we have $u \lhd u_i$ (since we have pumped the loop u_i, v_i, and u is still present in t'). By hypothesis, we have $r|_u = r|_v$. Let $\{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$ (resp. $\{j_1, \ldots, j_{k'}\} \subseteq \{1, \ldots, n\}$) be the maximal set of indices such that there is a k-ary context C_u (resp. k'-ary context C_v) such that $t|_u = C_u[t|_{u_{i_1}}, \ldots, t|_{u_{i_k}}]$ (resp. $t|_v = C_v[t|_{u_{j_1}}, \ldots, t|_{u_{j_{k'}}}]$). By definition of $\sim_{t,r}$, since $u \sim_{t,r} v$, we have $k = k'$, $C_u = C_v$, and $t|_{u_{i_l}} = t|_{u_{j_l}}$, for all $l \in \{1, \ldots, k\}$. By definition of the pumping, we have $t'|_u = C_u[t|_{v_{i_1}}, \ldots, t|_{v_{i_k}}]$ and $t'|_v = C_v[t|_{v_{j_1}}, \ldots, t|_{v_{j_k}}]$. Hence we get $t'_u = t'_v$, which ends up the proof. □

As a consequence, we have:

Corollary 1. *If $L(A) \neq \varnothing$, there is a tree $t \in L(A)$ whose height is bounded by $|Q|$.*

We can also use this technique to prove:

Lemma 3. *If there is a tree $t \in L(A)$ whose height is strictly greater than $|Q|$, then $L(A)$ is infinite.*

Proof. If there is a loop in a run involving a node u and a node v such that $u \lhd v$, we can iterate this loop in parallel below all nodes equivalent to v. Hence we keep the constraints satisfied. □

As a consequence of this lemma, we get this theorem (called Theorem 2 in the paper):

Theorem 7. *Let A be a positive TAGED. It is decidable whether $L(A)$ is infinite or not, in $O(|A||Q|^2)$ if $=_A \subseteq id_Q$, and in EXPTIME otherwise.*

Proof. If $=_A \subseteq id_Q$, we only have to test if there is a tree in $L(A)$ whose height is strictly greater than $|Q|$. This can be done by adding a counter to A, bounded by $|Q|+1$. Intuitively, if a tree t evaluates to (q,c), it means that the height of t is smaller than c. Let A' be the automaton we obtain. We let $(q,c) =_{A'} (q,c')$ iff $q =_A q$ and $c = c'$, so that we have $=_{A'} \subseteq id_{Q'}$. Emptiness is tested in linear time, as in the proof of Theorem 1.
If A does not satisfies $=_A \subseteq id_Q$, we first have to transform it, modulo an exponential blow-up, thanks to Lemma 1. □

C.2 Pumping Lemma for vbTAGED

Let (A,k) be a vbTAGED where $A = (\Sigma, Q, F, \Delta, =_A, \neq_A)$, such that $=_A \subseteq id_Q$. If $=_A \not\subseteq id_Q$, we first transform A into a TAGED A' such that $=_{A'} \subseteq id_{Q'}$ (thanks to Lemma 1), and we can prove that $L(A,k) = L(A',k)$.
Given two trees $t, t' \in T_\Sigma$, and a node $u \in N_t$, we denote by $t[u \leftarrow t']$ the tree t where the subtree at position u has been substituted by t'.
Let t be a tree and r a run of A on t which satisfies the equality constraints (but not necessarily all the disequality constraints). In the proof of Theorem 4, we use a pumping technique in order to repair the unsatisfied inequality constraints. Intuitively, t and r are repairable if one can substitute some subtrees (or subruns) of t and r in order to create a new tree t' together with a successful run of (A,k) on t'. More formally:

Definition 2. *Let t be a tree and r a run of A on t which satisfies the equality constraints, and such that $O_r(u_0)$ is a final state, where u_0 is the root node of t. We say that (t,r) is repairable if there are a set of nodes $\{u_1,\dots,u_n\} \subseteq N_t$, n trees $t_1,\dots,t_n \in T_\Sigma$, n runs $r_1,\dots,r_n$ of A on $t_1,\dots,t_n$ respectively, such that:*

1. *for all $1 \leq i,j \leq n$, if $i \neq j$, then u_i and u_j are incomparable by $\trianglelefteq$;*
2. *$r[u_1 \leftarrow r_1]\dots[u_n \leftarrow r_n]$ is a successful run of (A,k) on $t[u_1 \leftarrow t_1]\dots[u_n \leftarrow t_n]$.*

We say that (t',r') is a repair of (t,r).

In particular, in t, any constraint between two nodes $u, v \in N_t$, such that neither u nor v is comparable to some u_i, is satisfied.
Note that if there is a repairable pair (t,r), then $L(A,k)$ is non-empty. We give sufficient conditions for (t,r) to be repairable, and we can check in polynomial time whether those conditions hold. We prove that a pair (t,r) satisfying those conditions is repairable. We also prove that if $L(A,k)$ is non-empty, then there exists a pair (t,r) satisfying those conditions such that the height of t is bounded by $2(k+|\mathrm{dom}(=_A)|)|Q|$.
Actually, we prove a stronger result that imposes cardinality constraints on (t,r), so that if (t,r) is repairable, we can bound the size of a repair of (t,r).
In order to repair (t,r), we increase the size of some contexts of t and r. We introduce a notion of frontier below which all inequality constraints are satisfied. The repairing process goes inductively, by decreasing the frontier, for some order on frontiers. The proof is organized as follows:

1. we introduce technical background
2. we define sufficient conditions for a tree and a run to be repairable, through a predicate P;
3. we prove that a pair (t, r) satisfying those conditions is repairable (Theorem 8). The proof goes by induction on the frontiers. We state a base lemma (Lemma 4), and an induction lemma (Lemma 5);
4. we prove a lemma which states that if there is a tree in $L(A, k)$, then there is a tree and run of bounded size, depending on A and k only, that satisfy the sufficient conditions (Lemma 6);
5. we prove Theorem 4 and proves a corollary (Corollary 2) which states that we can bound the size of an accepted tree.

Contexts 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 a unary context C'_0 and contexts $C'_1, \ldots, C'_n$ such that n is the arity of C_1, and $C_2 = C'_0[C_1[C'_1, \ldots, C'_n]]$ (see [7] for a definition of contexts). Let $t \in L(A)$ 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, and 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 isomorphic to $\mathrm{cxt}_r(u)$ (we do not require the isomorphism to preserve the label relations).

Partial Orders Remind that $\sim_{t,r}$ was the equivalence relation introduced in Appendix A.4. Let $t \in L(A)$ and r a run of A on t which respects the equality constraints. For all nodes $u \in N_t$, we denote by $[u]$ its $\sim_{t,r}$-equivalence class. We define a strict partial order $\prec_{\sim_{t,r}}$ on equivalence classes by, for all nodes $u, v \in N_t$:

$$[u] \ \prec_{\sim_{t,r}} \ [v] \text{ iff } \exists u' \in [u],\ \exists v' \in [v],\ u' \lhd v'$$

$\prec_{\sim_{t,r}}$ is a strict partial order (see Lemma 7 at the end of the section). In particular, $\{u_0\}$ is the least class for $\prec_{\sim_{t,r}}$, where u_0 is the root of t. We denote by $\preceq_{\sim_{t,r}}$ its reflexive closure.
Now, we define a strict partial order $\prec^a_{\sim_{t,r}}$ on anti-chains of equivalence classes (i.e. sets of incomparable equivalence classes by $\prec_{\sim_{t,r}}$). The superscript a stands for "anti-chain". Let A, B be two anti-chains of equivalence classes. We let $A \prec^a_{\sim_{t,r}} B$ if $A \subsetneq B$ or the following hold: (i) for all $\alpha \in A$, there is $\beta \in B$ such that $\alpha \preceq_{\sim_{t,r}} \beta$, (ii) there is $\alpha \in A$ and $\beta \in B$ such that $\alpha \prec_{\sim_{t,r}} \beta$. Lemma 9 proves that $\prec^a_{\sim_{t,r}}$ is a strict partial order. Finally, for all anti-chain A of equivalence classes, we denote by $\bigcup A$ the set $\bigcup_{\alpha \in A} \alpha$. Note that $\bigcup A \subseteq N_t$.

Predicate Let $t \in T_\Sigma$, $r \in T_Q$, $F \subseteq 2^{N_t}$, $d, m \in \mathbb{N}$. We let $P(t, r, F, d, m)$ holds if all 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 $\mathrm{dom}(\neq_A)$;
2. for all $u, v \in N_r$, if $u \sim v$ then $r|_u = r|_v$;
3. $F \subseteq N_t/_{\sim_{t,r}}$ is an anti-chain of $\sim_{t,r}$-equivalence classes[8] (F is called the *frontier*);
4. $\bigcup F \subseteq \mathcal{C}(r)$;
5. if there are $u, v \in N_t$ such that $t|_u = t|_v$ and $O_r(u) \neq_A O_r(v)$, then there is $w \in \bigcup F$ such that either $u \trianglelefteq w$ or $v \trianglelefteq w$.
6. for all $v_1, v_2 \in N_t$, there is $u \in \mathcal{C}(r)$ such that $Rep(t, r, F, v_1, v_2, u)$ holds, meaning that if a disequality constraint between v_1 and v_2 is unsatisfied, we can repair it by increasing the size of the elementary context rooted at u. More formally, $Rep(t, r, F, v_1, v_2, u)$ is true if the following holds: if $t|_{v_1} = t|_{v_2}$ and $O_r(v_1) \neq_A O_r(v_2)$, then (i) either $v_1 \trianglelefteq u$ or $v_2 \trianglelefteq u$, (ii) there is $w \in \bigcup F$ such that $u \trianglelefteq w$, (iii) if $v_1 \trianglelefteq u$, then for all $u' \in N_t$ such that $v_2 \trianglelefteq u'$, if $\mathrm{path}_t(v_1, u)$ is isomorphic to $\mathrm{path}_t(v_2, u')$, then $u \not\sim_{t,r} u'$, (iv) if $v_2 \trianglelefteq u$, we define the condition symmetrically as (iii).
7. the length of any path from the root to any node of $\bigcup F$ is bounded by B, where $B = 2(k + |\mathrm{dom}(=_A)|)|Q|$;
8. The number of different subtrees of t which evaluates to inequality states in r is bounded by d: $|\{t' \mid \exists q \in \mathrm{dom}(\neq_A) \wedge \exists u \in N_t,\ t' = t|_u \wedge O_r(u) = q\}| \leq d$;
9. For all $u \in \mathcal{C}(r)$, if there is $v \in \bigcup F$ such that $u \trianglelefteq v$, then the height of $\mathrm{cxt}_t(u)$ is bounded by $2|Q|$;
10. the height of t is bounded by m.

Intuitions The frontier can be viewed as a set of incomparable nodes below which all inequality constraints are satisfied (condition 5). It is defined as classes of nodes which enroot elementary contexts, because we will use only elementary contexts to repair all the unsatisfied constraints which involve a node above the frontier. In particular, we will use loops contained in elementary contexts to increase their sizes. This can be done while keeping the equality constraints satisfied, because we change the contexts in parallel for all equivalent nodes. Condition 6 gives a sufficient condition to be able to repair an unsatisfied inequality constraint. The last four conditions imposes counting properties, in order to constrain the repairing process to go in a bounded way.
The next two lemmata are devoted to the proof of the following:

Theorem 8. *Let $t \in T_\Sigma$ and r a run of A on t which satisfies the equality constraints. If there are F, d, m such that $P(t, r, F, d, m)$ holds, then (t, r) is repairable, and there is a repair (t', r') of (t, r) such that the height of t' is bounded by $m + |Q|2^{m+3}(2^{2B+2} + d + 2^{m+B+3} + 2)$, where $B = 2(k + |dom(=_A)|)|Q|$.*

We start by a base lemma:

Lemma 4 (Base Lemma). *If $P(t, r, \varnothing, n, m)$ holds, then $t \in L(A, k)$.*

[8] $N_t/_{\sim_{t,r}}$ is the quotient space of N_t by $\sim_{t,r}$

Proof. This is due to conditions 1 and 5. □

We then prove an induction lemma, based on parallel pumping of elementary contexts:

Lemma 5 (Induction Lemma). *Let t, r, F, d, m such that $F \neq \varnothing$ and $P(t, r, F, d, m)$ holds, there are t', r', F' such that $P(t', r', F', d+2^{B+1}, m+2|Q|(2^{2B+2}+d+1))$ holds, $N_t \subseteq N_{t'}$ and $F' \prec^a_{\sim_{t,r}} F$, where $B = 2(k + |dom(=_A)|)|Q|$.*

Proof. **Construction of t', r', F'.**
Intuition. In order to obtain t' and r', we choose an equivalence class contained in the frontier F (it exists since $F \neq \varnothing$). Let $[u_1] = \{u_1, \ldots, u_n\}$ be this class. We let $D = \{v \in N_t \mid \exists i,\ v \trianglelefteq u_i\}$. There might be an unsatisfied inequality constraint between a node of t' and a node of D. By increasing the size of the elementary contexts rooted at $[u_1]$, we construct a tree t' together with a run r' such that there is no unsatisfied constraint between a node of t' and a node of D (except in one particular case that we explain further, but in this case, the constraints will be repaired further in the induction, thanks to condition 6). This can be done by using a loop contain in the elementary context rooted a u_1, and we do it in parallel for every u_i, so that we keep the equality constraints satisfied.
Parallel Growth of The Elementary Contexts. We first define formally how to increase the size of the contexts, while keeping the equality constraints satisfied. The idea is similar to the parallel pumping explain in Appendix C.1. 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 $O_r(\alpha_1) = O_r(\beta_1) = q$. For all $i \in \{2, \ldots, n\}$, we define α_i as the node below u_i such that $\mathrm{path}_t(u_i, \alpha_i)$ is isomorphic to $\mathrm{path}_t(u_1, \alpha_1)$ (it exists since $u_1 \sim_{t,r} u_i$ and $t|_{u_1} = t|_{u_i}$ by Prop. 8). Note that by Prop. 9, $\alpha_1 \sim_{t,r} \alpha_i$. We define the nodes β_i similarly, and also get $\beta_1 \sim_{t,r} \beta_i$. By condition 2, for all $i \in \{1, \ldots, n\}$, $O_r(\alpha_i) = O_r(\beta_i) = q$. 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 $[u_1]$, and by definition of $\sim_{r,t}$, the equality constraints are still satisfied by r^j on t^j, for all $j \in \mathbb{N}$. This pumping is illustrated in Fig. 6.
Existence of a Repairing Run We first define the set of pairs of nodes $(u, v) \in N_t \times N_t$ which can possibly give an unsatisfied inequality constraint when pumping, but for which there is a way to pump such that the constraint is satisfied. These pairs are called candidates. We denote by $Cand([u_1])$ this set. For all $u, v \in N_t$, $(u, v) \in Cand([u_1])$ if (i) $u \in \uparrow[u_1]$[9], (ii) $O_r(u) \neq_A O_r(v)$, (iii) if for all $j \in \{1, \ldots, n\}$, $Rep(t, r, F, u, v, u_j)$ does not hold, then $t|_u \neq t|_v$.

[9] $\uparrow[u_1] = \{w \mid \exists w' \in [u_1],\ w \trianglelefteq w'\}$

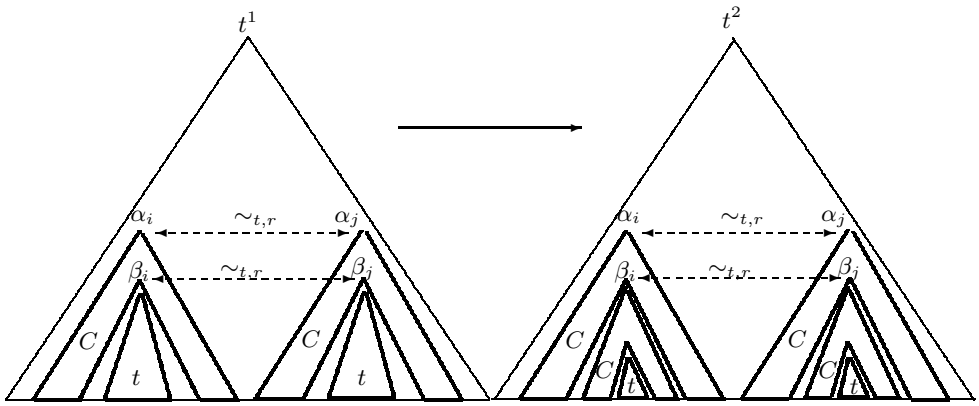

Fig. 6. parallel growth in elementary contexts

Note that by definition of the pumping, for all $i > 0$, for all nodes $u \in N_t$ such that $O_r(u) \in \mathrm{dom}(=_A) \cup \mathrm{dom}(\neq_A)$, u is still a node of t^i. For all $i > 0$, and all pairs $(u, v) \in Cand([u_1])$, we say that i is *incompatible* with (u, v) if $t^i|_u = t^i|_v$.
Claim 1. *For all pairs $(u, v) \in Cand([u_1])$, there is at most one $i > 0$ such that i is incompatible with (u, v)*

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

- $v \notin \uparrow[u_1]$. We have $O_{r^i}(v) \neq_A O_{r^i}(u)$, $t^i|_v = t^i|_u$, $O_{r^j}(v) \neq_A O_{r^j}(u)$ and $t^j|_v = t^j|_u$. Since $O_r(v) \in\in \mathrm{dom}(\neq_A)$, and $v \notin \uparrow[u_1]$, v is below, or incomparable, to any node which belongs to an elementary context rooted at a node of $[u_1]$. 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 $u \in \uparrow[u_1]$, by definition of the pumping, we have $|t^j|_u| > |t^i|_u|$, which contradicts $t^j|_u = t^i|_u$;
- $v \in \uparrow[u_1]$. We consider two cases:
  - if there is $\ell \in \{1, \ldots, n\}$ such that $Rep(t, r, F, u, v, u_\ell)$ holds. By definition of Rep, we have $t|_u = t|_v$. Suppose that $u \trianglelefteq u_\ell$ and let v_ℓ be the node below v such that $\mathrm{path}_t(u, u_\ell)$ is isomorphic to $\mathrm{path}_t(v, v_\ell)$ (it exists since $t|_u = t|_v$). By definition of the predicate Rep, $u_\ell \not\sim_{t,r} v_\ell$. Since $\mathrm{path}_t(u, u_\ell)$ and $\mathrm{path}_t(v, v_\ell)$ are isomorphic, and $t|_u = t|_v$, we also get $t|_{u_\ell} = t|_{v_\ell}$. This implies that there is no node of $[u_1]$ comparable to v_ℓ (by $\trianglelefteq$): suppose that there is some node $u_i \sim_{t,r} u_1$ comparable to v_ℓ such that $u_i \neq v_\ell$. By Prop. 8, we have $t|_{u_i} = t|_{u_1}$, which contradicts $t|_{u_1} = t|_{v_\ell}$.
    Hence the subtree at position v_ℓ does not change during the pumping. In particular, $t^i|_{v_\ell} = t^j|_{v_\ell}$. Since $u_\ell \in [u_1]$, we pump below u_ℓ. Hence we have $t^i|_{u_\ell} \neq t^j|_{u_\ell}$. Therefore, either $t^i|_{u_\ell} \neq t^i|_{v_\ell}$ or $t^j|_{u_\ell} \neq t^j|_{v_\ell}$.

Since $\text{path}_t(u, u_\ell)$ and $\text{path}_t(v, v_\ell)$ are isomorphic, and there is no node of $[u_1]$ comparable to v_ℓ, by definition of the pumping, we also have that $\text{path}_{t^i}(u, u_\ell)$ and $\text{path}_{t^i}(v, v_\ell)$ are isomorphic, and $\text{path}_{t^j}(u, u_\ell)$ and $\text{path}_{t^j}(v, v_\ell)$ are isomorphic. Therefore, either $t^i|_v \neq t^i|_u$ or $t^j|_v \neq t^j|_u$;

- If for all $\ell \in \{1, \ldots, n\}$, $Rep(t, r, F, u, v, u_\ell)$ does not hold. By definition of $Cand([u_1])$, it implies that $t|_u \neq t|_v$. By hypothesis, $t^i|_u = t^i|_v$. We first prove that there is necessarily $\ell \in \{1, \ldots, n\}$ such that $Rep(t^i, r^i, F, u, v, u_\ell)$. Suppose the contrary. It means for all $\ell \in \{1, \ldots, n\}$ such that $u \trianglelefteq u_\ell$, and for all v' such that $\text{path}_{t^i}(u, u_\ell)$ is isomorphic to $\text{path}_{t^i}(v, v')$, we have $v' \sim_{t,r} u_\ell$. And symmetrically, for all $\ell \in \{1, \ldots, n\}$ such that $v \trianglelefteq u_\ell$, and for all u' such that $\text{path}_{t^i}(u, u')$ is isomorphic to $\text{path}_{t^i}(v, u_\ell)$, we have $u' \sim_{t,r} v_\ell$. Since $t|_u \neq t|_v$, and thanks to Lemma 10, this inequality is preserved during pumping. Hence $t^i|_u \neq t^i|_v$, which contradicts $t^i|_u = t^i|_v$.
  Hence, there is $\ell \in \{1, \ldots, n\}$ such that $Rep(t^i, r^i, F, u, v, u_\ell)$. Suppose that $u \trianglelefteq u_\ell$ (the case $v \trianglelefteq u_\ell$ is symmetric). Let v' such that $v \trianglelefteq v'$ and $\text{path}_{t^i}(v, v')$ is isomorphic to $\text{path}_{t^i}(u, u_\ell)$. By definition of Rep, we have $u_\ell \not\sim_{t^i, r^i} v'$. Since $t^i|_u = t^i|_v$, we also get $t^i|_{u_\ell} = t^i|_{v'}$. For the same reasons than the previous case, there is no node $w \in [u_1]$ which is comparable to v'. Hence, since we pump only in the elementary contexts rooted at nodes of $[u_1]$, the subtree rooted at v' does not change during pumping. Since the subtree rooted at u_ℓ changes during pumping and $\text{path}_{t^i}(u, u_\ell)$ is isomorphic to $\text{path}_{t^i}(v, v')$, we necessarily have $t^j|_u \neq t^j|_v$, which contradicts the hypothesis.

End of Proof of Claim 1 $\square$.

Hence, we can define a function $\Psi : Cand([u_1]) \to \mathbb{N} - \{0\}$ such that $\Psi(u, v)$ is the unique index (if it exists) i incompatible with (u, v), for all $(u, v) \in Cand([u_1])$. Now, we prove that we can bound the size of the range of Ψ. Let $\overline{\uparrow[u_1]}$ be the set of nodes u such that $u \notin \uparrow[u_1]$. For all $u \in \uparrow[u_1]$, and all $v, v' \in \overline{\uparrow[u_1]}$ such that $(u, v) \in Cand([u_1])$, $(u, v') \in Cand([u_1])$, if $t|_v = t|_{v'}$, then $\Psi(u, v) = \Psi(u, v')$ (if defined). By condition 8, we have $|\{t|_v \mid v \in \overline{\uparrow[u_1]}\}| \leq d$, hence there are at most d elements of $\mathbb{N} - \{0\}$ whose pre-image are in $\overline{\uparrow[u_1]}$. In other hand, by condition 7, we have $|\uparrow[u_1] \times \uparrow[u_1]| \leq 2^{2B+2}$. Hence, the size of the range of Ψ is bounded by $d + 2^{2B+2}$. Hence, there is $i_0 \in \{1, \ldots, d + 2^{2B+2} + 1\}$ such that i_0 has no pre-image by Ψ. We define t' by t^{i_0} and r' by r^{i_0}. Note that by definition of the pumping, condition 2 still holds for $\sim_{t', r'}$, since we make the elementary contexts grow in parallel below all equivalent nodes of $[u_1]$, 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 do not increase the number of nodes ordered by $\lhd$, and labeled in $\text{dom}(\neq_A)$. Hence condition 1 holds for r'.

Remark 1 By definition of the pumping, all inequality constraints between pairs of nodes $(u, v) \in Cand([u_1])$ are satisfied in r^{i_0}.

We now define F'. Let V be the set of greatest class (for $\prec_{\sim_{t,r}}$) strictly lesser than $[u_1]$, and such that for all $\alpha \in V$, there is no node $\beta \in F - [u_1]$ such that

$\alpha \prec_{\sim_{t,r}} \beta$, and for all $u \in \alpha$, $u \in \mathcal{C}(r)$. We let $F' = (F - [u_1]) \cup V$. Note that $F' \prec^a_{\sim_{t,r}} F$ and F' satisfies condition 4. Thanks to condition 2, for all u, v such that $u \sim_{t',r'} v$, if $u \in \mathcal{C}(r')$, then $v \in \mathcal{C}(r')$. It implies that F' satisfies condition 3, i.e. $F' \subseteq N_{t'}/_{\sim_{t',r'}}$.
Before proving that $P(t', r', F', d+2^{B+1}, m+2|Q|(2^{2B+2}+d+1))$ holds, we first prove a useful claim, which states that all inequality constraints below $\bigcup F'$ are satisfied:
Claim 2. *For all $u, v \in N_{t'}$, such that $u, v \notin \uparrow(\bigcup F')$, and $O_{r'}(u) \neq_A O_{r'}(v)$, we have $t'|_u \neq t'|_v$*

Proof. We start by the following fact (*): by definition of $\prec_{\sim_{t,r}}$, all nodes of $\bigcup F'$ are above the nodes of $\bigcup F$ in t. We now consider two cases:

- $u, v \notin \uparrow[u_1]$. Since $O_r(u), O_r(v) \in \mathrm{dom}(\neq_A)$, neither u nor v belongs to the elementary contexts rooted at nodes of $[u_1]$. Hence the subtrees at positions u and v have not changed during the pumping. Hence, thanks to condition 5, and to the fact (*), we have $t|_u \neq t|_v$ and still have $t'|_u \neq t'|_v$;
- if $u \in \uparrow[u_1]$. Thanks to remark 1, we have to prove that $(u, v) \in Cand([u_1])$. We only prove point (iii) of the definition of $Cand$. Suppose that for all $i \in \{1, \ldots, n\}$, $Rep(t, r, F, u, v, u_i)$ does not hold, and $t|_u = t|_v$. Let $i \in \{1, \ldots, n\}$, and suppose that $u \trianglelefteq u_i$. Since $Rep(t, r, F, u, v, u_i)$ does not hold, for all v' such that $\mathrm{path}_t(u, u_i)$ is isomorphic to $\mathrm{path}_t(v, v')$, we necessarily have $u_i \sim_{t,r} v'$. From this we can deduce that $v \in \uparrow[u_1]$. By condition 6, there is $w \in \mathcal{C}(r)$, such that $Rep(t, r, F, u, v, w)$ holds. Hence w is above $\bigcup F$, by definition of Rep. Moreover, $w \notin [u_1]$ (by hypothesis). Hence, either v or u is above a node of $\bigcup F'$, by definition of F', which is a contradiction.

End of Proof of Claim 2 □.

Correctness We now prove that $P(t', r', F', d+2^{B+1}, m+2|Q|(2^{2B+2}+d+1))$ holds, i.e. every condition is satisfied:

- Condition 1, 2, 3, and 4 have already been proved;
- Condition 5. This is proved in Claim 2.
- Condition 6. Let $v_1, v_2 \in N_{t'}$ such that $t'|_{v_1} = t'|_{v_2}$ and $O_{r'}(v_1) \neq_A O_{r'}(v_2)$. We consider several cases:
  - $v_1 \notin \uparrow[u_1]$ and $v_2 \notin \uparrow[u_1]$. Since $O_r(v_1), O_r(v_2) \in \mathrm{dom}(\neq_A)$, neither v_1 nor v_2 are in the elementary contexts rooted at the nodes of $[u_1]$. Hence the subtrees at positions v_1 and v_2 have not changed during pumping. Since $P(t, r, F, d, m)$ holds, there is a node $u \in \mathcal{C}(r)$ such that u is above $\bigcup F$ and $Rep(t, r, F, v_1, v_2, u)$ holds. The node u is necessarily above $\bigcup F'$. Otherwise it would mean that $u \in [u_1]$, which would contradict $v_1 \notin \uparrow[u_1]$ or would contradict $v_2 \notin \uparrow[u_1]$. Hence we also get that $Rep(t', r', F', v_1, v_2, u)$ holds;
  - $v_1 \in \uparrow[u_1]$. In this case, $(v_1, v_2) \notin Cand([u_1])$, otherwise $t'|_{v_1} \neq t'|_{v_2}$. By definition of $Cand([u_1])$, $t|_{v_1} = t|_{v_2}$ and for all $i \in \{1, \ldots, n\}$, $Rep(t, r, v_1, v_2, u_i)$ does not hold. By condition 6 of $P(t, r, F, d, m)$, there is a node $u \in \mathcal{C}(r)$

such that u is above $\bigcup F$ and $Rep(t, r, F, v_1, v_2, u)$ holds. Hence $u \notin [u_1]$. Since u is above $\bigcup F$, we get that u is also above $\bigcup F'$. Hence $Rep(t', r', F', v_1, v_2, u)$ holds.
  - the last case is symmetric to the latter.
- Condition 7. The new frontier F' is smaller than F and thus, nodes from $\bigcup F'$ are closer to the root than those of $\bigcup F$
- Condition 8. We create at most as many different subtrees as there are nodes above $\bigcup F$. By condition 7, there are at most 2^{B+1} nodes above $\bigcup F$.
- Condition 9. Since we have not changed the elementary contexts above $\bigcup F$, and thanks to Lemma 8, this condition still holds.
- Condition 10. First not that the height of $\mathrm{cxt}_t(u_i)$ is bounded by $2|Q|$ for all $i \in \{1, \ldots, n\}$ (thanks to condition 9). Since $i_0 \leq d + 2^{2B+2} + 1$, the height of $t^{i_0} = t'$ is bounded by the height of t plus $2|Q|(2^{2B+2} + d + 1)$, i.e. $m + 2|Q|(2^{2B+2} + d + 1)$.

End of proof of Lemma 5 □

Proof (Proof of Theorem 8). Suppose that there is F, d, m such that $P(t, r, F, d, m)$ holds. Hence there are at most 2^{m+1} nodes u of r such that $O_r(u) \in \mathrm{dom}(=_A) \cup \mathrm{dom}(\neq_A)$. Let $F_0 \prec^a_{\sim_{t,r}} F_1 \prec^a_{\sim_{t,r}} \ldots \prec^a_{\sim_{t,r}} F_p$ be a maximal chain of frontiers (in particular, for all $i \in \{0, \ldots, p\}$, $F_i \subseteq N_t/_{\sim_{t,r}}$), where $F_0 = \varnothing$ and $F_p = F$. For every node $u \in N_t$ such that $O_r(u) \in \mathrm{dom}(=_A) \cup \mathrm{dom}(\neq_A)$, there is at most one $i \in \{1, \ldots, p\}$ such that $[u]$ is removed from F_{i-1} to get F_i, and at most one $i \in \{1, \ldots, p\}$ such that $[u]$ is added to F_i (and was not in F_{i-1}). Since there are at most 2^{m+1} such classes, it implies that $p \leq 2^{m+2}$.
Hence, from Lemma 5, there are t', r', d', m' such that $P(t', r', \varnothing, d', m')$. Moreover, we have $d' \leq d + 2^{m+2}2^{B+1}$, and $m' \leq m + 2|Q|2^{m+2}(2^{2B+2} + d' + 1)$, i.e. $m' \leq m + |Q|2^{m+3}(2^{2B+2} + d + 2^{m+B+3} + 1)$.
By Lemma 4, r' is a successful run of (A, k) on r'. Since the parallel pumping does nothing else than substituting subtrees rooted at nodes labeled by $\mathrm{dom}(=_A) \cup \mathrm{dom}(\neq_A)$ in r, and (t', r') is a repair of (t, r). □

Lemma 6. *We let $B = 2(k + |dom(=_A)|)|Q|$, and $B' = (k + 2^{|Q|})2^{|Q|+1}$.*
If $L(A, k) \neq \varnothing$, there is a tree t, a run r of A on t, some natural d and some frontier F such that $P(t, r, F, d, B)$ holds and the height of t is bounded by B (and by B' if $=_A \not\subseteq id_Q$).

Proof. We say that F is the maximal frontier of t, r if it is a maximal anti-chain (for $\prec^a_{\sim_{t,r}}$ which satisfies conditions 3 and 4. We can prove it is unique, and equal to the set of $\sim_{t,r}$-classes $[u]$ such that u is a maximal node for $\lhd$ such that $u \in \mathcal{C}(r)$.
We let $P'(t, r, F)$ the relaxed predicate P where we only consider conditions 1, 2, 3, 4, 5, and 6. Given an elementary context C in r, we say that C contains a 3-loop if there are three descendant nodes of C which are labeled by the same state.
Claim 1. *For all t, r, F, if $P'(t, r, F)$ holds, F is the maximal frontier of t, r and there is an elementary context of r which contains a 3-loop, then there are*

t', r', F' such that $P(t', r', F')$, F' is the maximal frontier of t', r', and the size of t' is strictly lesser than the size of t.

Proof. We use a similar technique as the pumping technique presented in the proof of Lemma 2. But, instead of pumping maximally, we pump the 3-loops in parallel in elementary contexts which contain a 3-loop. In each elementary context, we pump the two greatest nodes of the 3-loop. This ensures that there is still a loop in the elementary context after pumping.
First note that by hypothesis, $\mathcal{C}(r)$ is non-empty, and there is $u \in \mathcal{C}(r)$ such that $\mathrm{cxt}_r(u)$ contains a 3-loop. We let $[u]$ be its equivalence class by $\sim_{t,r}$. By hypothesis, we have $r|_v = r|_u$, for all $v \in [u]$. Hence $v \in \mathcal{C}(r)$ for all $v \in [u]$. Again by definition of $\sim_{t,r}$, all the nodes of $[u]$ are incomparable. Let $n = |[u]|$ and $\{u_1, \ldots, u_n\} = [u]$. We can define C 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 $O_r(\alpha_i) = O_r(\beta_i) = O_r(\gamma_i) = q$. Moreover, we take $\alpha_i, \beta_i, \gamma_i$ such that for all $i, j \in \{1, \ldots, n\}$, we have $\mathrm{path}_t(u_i, \alpha_i)$ isomorphic to $\mathrm{path}_t(u_j, \alpha_j)$, $\mathrm{path}_t(\alpha_i, \beta_i)$ isomorphic to $\mathrm{path}_t(\alpha_j, \beta_j)$, and $\mathrm{path}_t(\beta_i, \gamma_i)$ isomorphic to $\mathrm{path}_t(\beta_j, \gamma_j)$. 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. 7 (the subtrees $t|_{\gamma_i}$ are represented in dashed style). It is technical but not difficult to prove that r' is a run of A on t' such that its root is labeled by a final state and it respects the equality constraints. We let F' be the maximal frontier of t', r'.
Fact 1. $N_{t'} \subseteq N_t$ and for all $v_1, v_2 \in N_{t'}$, $v_1 \sim_{t',r'} v_2$ iff $v_1 \sim_{t,r} v_2$. Its proof is given at the end of Subsection C.3.
We prove that conditions 1, 2, 3, 4, 5, and 6 hold for t',r', and F'.

- conditions 1, 2. The equality constraints are still satisfied since we pump in parallel below equivalent nodes. The other conditions obviously hold;
- condition 3 and 4. This is by definition of F'.
- condition 5. Since we only remove nodes of t to get t', we have $N_{t'} \subsetneq N_t$. Hence $N_{r'} \subsetneq N_r$. By definition of the pumping, we also have $\mathcal{C}(r') \subseteq \mathcal{C}(r)$.
  Let $v \notin \uparrow_{r'}(\bigcup F')$ such that $O_{r'}(v) \in \mathrm{dom}(=_A) \cup \mathrm{dom}(\neq_A)$. We prove that $v \notin \uparrow_r(\bigcup F)$. Suppose that there is $w \in \bigcup F$ such that $v \trianglelefteq w$ in r. Suppose that $w \in N_{t'}$. We then have $v \trianglelefteq w$ in r', and $w \in \mathcal{C}(r')$. Hence $w \notin \uparrow_{r'}(\bigcup F')$ and it contradicts maximality of F'. Hence $w \notin N_{t'}$. Hence w has been removed from t due to some pumping in an elementary context rooted at some node $v' \in \mathcal{C}(r)$. By definition of the pumping, we have $v'\mathcal{C}(r')$. If $v \trianglelefteq v'$, since $v \notin \uparrow_{r'}(\bigcup F')$, it contradicts maximality of F'. Hence $v' \lhd v$. Since $O_{r'}(v) \in \mathrm{dom}(=_A) \cup \mathrm{dom}(\neq_A)$, we have $v \notin N_{\mathrm{cxt}_{r'}(v')}$. Moreover, $v \trianglelefteq w$. Since w has been removed by the pumping of $\mathrm{cxt}_{r'}(v')$, and $v' \lhd v \trianglelefteq w$, and $v \notin N_{\mathrm{cxt}_{r'}(v')}$, v has necessarily been removed from t by the pumping. This contradicts $v \in N_{r'}$. This conclude the proof that $v \notin \uparrow_r(\bigcup F)$.
  Let now $v_1, v_2 \in N_{t'}$ such that $v_1, v_2 \notin \uparrow_{r'}(\bigcup F')$, and $O_{r'}(v_1) \neq_A O_{r'}(v_2)$. Hence $v_1, v_2 \notin \uparrow_r(\bigcup F)$. Hence the subtrees rooted at nodes v_1 and v_2 respectively have not changed during pumping, i.e. $t|_{v_1} = t'|_{v_1}$, and $t|_{v_2} = t'|_{v_2}$.

Since condition 5 is satisfied by t, r, F, we get $t'_{v_1} \neq t'_{v_2}$, which conclude the proof.

– condition 6. Let $v_1, v_2 \in N_{t'}$ such that $O_{r'}(v_1) \neq_A O_{r'}(v_2)$ and $t'|_{v_1} = t'|_{v_2}$. Since condition 5 is satisfied, either v_1 or v_2 is above a node of $\bigcup F'$. We consider two cases:
  - if $t|_{v_1} \neq t|_{v_2}$, then it means that the pumping has "broken" the inequality constraint at positions v_1 and v_2. Let $[u] = \{u_1, \ldots, u_n\}$ be the nodes defined in the definition of the pumping, i.e. the nodes which enroot elementary contexts in which we have pumped. Let $\{u_{i_1}, \ldots, u_{i_{n_1}}\} \subseteq [u]$ (resp. $\{u_{j_1}, \ldots, u_{j_{n_2}} \subseteq [u]$) the nodes of $[u]$ which are below v_1 (resp. v_2). Note that by definition of the pumping, we still have $[u] \subseteq N_{t'}$, and the context of t which have nodes of $[u]$ as holes has not changed during pumping. Moreover, the $\sim_{t,r}$-equivalence class of u is equal to the $\sim_{t',r'}$-equivalence class of u. Hence, for all $\ell \in \{1, \ldots, n_1\}$, we can define the node u'_{i_ℓ} of $N_{t'}$ such that $\text{path}_{t'}(v_1, u_{i_\ell})$ is isomorphic to $\text{path}_{t'}(v_2, u'_{i_\ell})$. Similarly, for all $\ell \in \{1, \ldots, n_2\}$, we can define u'_{j_ℓ} the node below v_1 such that $\text{path}_{t'}(v_2, u_{j_\ell})$ is isomorphic to $\text{path}_{t'}(v_1, u'_{j_\ell})$. Suppose that for all $\ell \in \{1, \ldots, n_1\}$, we have $u_{i_\ell} \sim_{t',r'} u'_{i_\ell}$, and for all $\ell \in \{1, \ldots, n_2\}$, we have $u_{j_\ell} \sim_{t',r'} u'_{j_\ell}$. By Lemma 10, since $t|_{v_1} \neq t|_{v_2}$, we necessarily have $t'|_{v_1} \neq t'_{v_2}$, which contradicts $t'|_{v_1} = t'_{v_2}$. Hence there is $\ell_1 \in \{1, \ldots, n_1\}$ or $\ell_2 \in \{1, \ldots, n_2\}$ such that $u_{i_{\ell_1}} \not\sim_{t',r'} u'_{i_{\ell_1}}$ or $u_{j_{\ell_2}} \not\sim_{t',r'} u'_{j_{\ell_2}}$. Suppose it is $u_{i_{\ell_1}}$: $u_{i_{\ell_1}}$ is above F', so that $\mathit{Rep}(t', r', F', v_1, v_2, u_{i_{\ell_1}})$ holds;
  - if $t|_{v_1} = t|_{v_2}$. Since condition 6 holds for t, r, F, there is $v \in \mathcal{C}(r)$ such that $\mathit{Rep}(t, r, F, v_1, v_2, v)$ holds. Suppose that $v_1 \trianglelefteq v$ in t, the other case being symmetric. Let v' be the node of below v_2 such that $\text{path}_t(v_1, v)$ is isomorphic to $\text{path}_t(v_2, v')$. By definition of the predicate Rep, $v \not\sim_{t,r} v'$. Let $[u]$ be the class as defined in the definition of the pumping. We now consider three cases:
    * Suppose there is a node $w \in [u]$ such that $v_1 \leq w \leq v$, and let w' below v_2 such that $\text{path}_t(v_1, w)$ is isomorphic to $\text{path}_t(v_2, w')$. If $w \sim_{t,r} w'$, then by definition of $\sim_{t,r}$, 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 $[u]$, $w \in N_{t'}$. Since $w \in \mathcal{C}(r')$ and F' is maximal, w is above a node of $\bigcup F'$. Hence $\mathit{Rep}(t', r', F', v_1, v_2, w)$ holds;
    * Suppose there is a node $w \in [u]$ such that $v_2 \leq w \leq v'$. With exactly the same arguments we can prove that $\mathit{Rep}(t', r', F', v_1, v_2, w)$ holds;
    * Suppose that there is no node $w \in [u]$ such that $v_1 \trianglelefteq w \trianglelefteq v$ or $v_2 \trianglelefteq w \trianglelefteq v'$. Since we pump below nodes of $[u]$, and v_1, v_2 are still in $N_{t'}$, $\text{path}_{t'}(v_1, v)$ is isomorphic to $\text{path}_{t'}(v_2, v')$. Moreover, $v \in \mathcal{C}(r')$, so that $v \in \uparrow_{r'} (\bigcup F')$. Finally, thanks to Fact 1, since $v \not\sim_{t,r} v'$, we also get $v \not\sim_{t',r'} v'$. Hence $\mathit{Rep}(t', r', F', v_1, v_2, v)$ holds.

  Finally, it is clear that the size of t' is strictly lesser than the size of t.

End of Proof of Claim 1 □

Let $B = 2(k + |\text{dom}(=_A)|)|Q|$. If $L(A, k) \neq \varnothing$, there is $t \in L(A)$ and r a successful run of A on t. Let C be the maximal context of r (for context inclusion) such

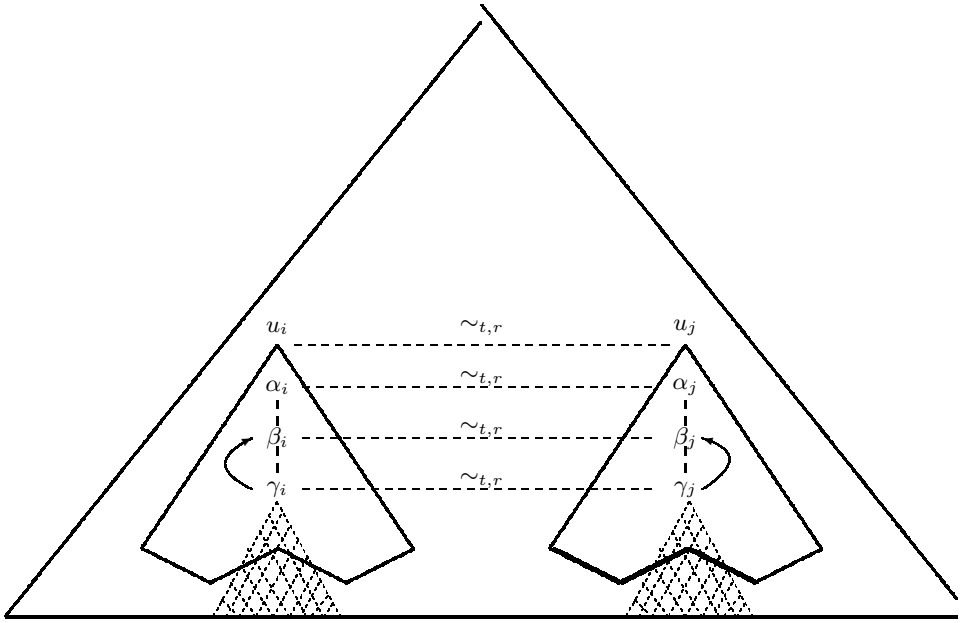

Fig. 7. parallel pumping in elementary contexts

that the root of r is the root of C, and no nodes of C (except possibly the root) are labeled in $\mathrm{dom}(=_A) \cup \mathrm{dom}(\neq_A)$. We call this context the *top context*. If the root of C is not labeled by a state of $\mathrm{dom}(=_A) \cup \mathrm{dom}(\neq_A)$, we first pump in C, while there is a loop. We do the corresponding pumping in t. Note that all this pumping preserves the satisfiability of the constraints, since we pump above the nodes where a test occurs. We obtain a successful run r' of (A, k) on a tree t' the such that the height of the top context of r' is at most $|Q|$ if the root of C is not labeled in $\mathrm{dom}(=_A) \cup \mathrm{dom}(\neq_A)$. We call this property P_1. We let F' be the maximal frontier of r'. It is not difficult to prove that $P'(t', r', F')$ holds.
Now, 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 $|\mathrm{dom}(=_A)|$ 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). Thanks to property P_1, it implies that there is an elementary context containing a 3-loop. Hence all the conditions to apply the rewriting of Claim 1 are satisfied, and we do it while there is an elementary context containing a 3-loop. At the end, we get a tree t'', a run r'' and a frontier F'' such that the height of t'' is bounded by B, and $P'(t'', r'', F'')$ holds. Hence, t'' have at most 2^{B+1} nodes, and every elementary context of r'' have its height bounded by $2|Q|$ (otherwise there would be a 3-loop). Hence $P(t'', r'', F'', 2^{B+1}, B)$ holds.
If the height of t is lesser than B, since we already have $P'(t, r, F)$, and the number of nodes of t is lesser than 2^{B+1}, $P(t, r, F, 2^{B+1}, B)$ holds.

If A does not satisfy $=_A \subseteq id_Q$, we first have to transform it (modulo an exponential blow-up). $\square$

We now have the following corollary:

Corollary 2. *If $L(A,k) \neq \varnothing$, then there is a tree $t \in L(A,k)$ whose height is bounded by $B + |Q|2^{3B+8}$ (and by $|Q|2^{2^{|Q|+3}(k+2^{|Q|+1})}$ if $=_A \not\subseteq id_Q$), where $B = 2(k + |dom(=_A)|)|Q|$.*

Proof. The proof of Lemma 6 proves that there is t', r', F, d such that $P(t', r', F, d, B)$ holds. Moreover, we necessarily have $d \leq 2^{B+1}$. Hence $P(t', r', F, 2^{B+1}, B)$ holds. By Theorem 8, there is $t \in L(A,k)$ whose height is bounded by $B+|Q|2^{B+3}(2^{2B+2}+ d + 2^{2B+3} + 1)$. We can prove that the height of t is therefore bounded by $|Q|2^{10k|Q|+6}$.
If A does not satisfy $=_A \subseteq id_Q$, we first have to transform it (modulo an exponential blow-up), and in this case we can bound the height of an accepted tree by $|Q|2^{2^{|Q|+3}(k+2^{|Q|+1})}$. $\square$

Proof of Theorem 4 Suppose that $=_A \subseteq id_Q$. By Lemma 6 and Theorem 8, it suffices to guess a pair (t, r) such that the height of t (and r) is bounded by $B = 2(k + |Q|)|Q|$, and such that there are F, d, m such that $P(t, r, F, d, m)$ holds. Moreover, by inspecting the proof of Lemma 6, we can fix F to be the maximal frontier of (t, r), and we can take $d = 2^{B+1}$, and $m = B$. Moreover, we can check in PTIME (in $|t|$, $|r|$, $|F|$ – which is lesser than $|t|$ –, $log_2(d)$, and m) that $P(t, r, F, d, m)$ holds.
The size of the tree is at most $2^{2(k+|Q|)|Q|+1}$, which is doubly exponential in the size of the binary encoding of k.
If $=_A \not\subseteq A$, the height of the tree (and the run) we have to guess is at most $2^{(k+2^{|Q|})2^{|Q|+1}+1}$, which is doubly exponential both in the size of $|Q|$, and in the size of the binary encoding of k. $\square$

C.3 Minor Lemmata

Lemma 7. *For all tree $t \in L(A)$ and all run r of A on t which respects the equality constraints, $\prec_\sim$ is a strict partial order on $\sim$-equivalence classes.*

Proof. Remind that for all $u, v \in N_t$, if $u \sim v$, then $t|_u = t|_v$.

- it is irreflexive. Suppose that there is $u \in N_t$ such that $[u] \prec_\sim [u]$. Hence, there is $v \in [u]$, and $w \in [u]$ such that $v \lhd w$, which contradicts $t|_v = t|_w$;
- it is transitive. Let $u, v, w \in N_t$ such that $[u] \prec_\sim [v]$ and $[v] \prec_\sim [w]$. Hence, there are $u' \in [u]$ and $v' \in [v]$ such that $u' \lhd v'$, and there are $v'' \in [v]$ and $w'' \in [w]$ such that $v'' \lhd w''$. Let $w' \in N_t$ such that $v' \lhd w'$ and $\text{path}_t(v', w')$ is isomorphic to $\text{path}_t(v'', w'')$ (it exists since $t|_{v'} = t|_{v''}$)). Since $u' \lhd w''$, we get $[u] \prec_\sim [w]$.
- it is asymmetric, because it is irreflexive and transitive.

□

Lemma 8. *Let $t \in L(A)$ and r a run of A on t satisfying the equality constraints. Let $u, v \in N_t$ such that $[u] \prec_{\sim} [v]$ (where $[u]$ and $[v]$ are the $\sim$ equivalence classes of u and v respectively). Then for all $u' \in [u]$, there is $v' \in [v]$ such that $u' \lhd v'$.*

Proof. Let $u' \in [u]$. By definition of $\prec_{\sim}$, there are $u_0 \in [u]$ and $v_0 \in [v]$ such that $u_0 \lhd v_0$. Since $u' \sim u_0$, we have $t|_{u'} = t|_{u_0}$. We let $v' \in N_t$ such that $u' \lhd v'$ and $\text{path}_t(u', v')$ is isomorphic to $\text{path}_t(u_0, v_0)$. By Prop. 9 of $\sim$, we have $v' \sim v$, which conclude the proof. □

Lemma 9. *Let $t \in L(A)$ and r a run of A on t satisfying the equality constraints. $\prec^a_{\sim_{t,r}}$ is a strict partial order on anti-chains of $\sim_{t,r}$-equivalence classes.*

Proof.
- *it is irreflexive.* Suppose it is reflexive, and there is some A such that $A \prec^a_{\sim_{t,r}} A$. By definition there is $a \in A$ and $b \in A$ such that $a \prec_{\sim_{t,r}} b$, which contradicts the fact that A is an anti-chain;
- *it is transitive.* Let A, B, C such that $A \prec^a_{\sim_{t,r}} B$ and $B \prec^a_{\sim_{t,r}} C$. We have to consider several cases:
  - if $A \subsetneq B$ and $B \subsetneq C$, then $A \subsetneq C$;
  - if $A \subsetneq B$ and $B \not\subseteq C$. For all $a \in A$, we have $a \in B$, and there is $c \in C$ such that $a \preceq_{\sim_{t,r}} c$. If there is $a \in A$ and $c \in C$ such that $a \prec_{\sim_{t,r}} c$, then we get $A \prec^a_{\sim_{t,r}} C$. Otherwise, it means that for all $a \in A$ and all $c \in C$, either a and c are incomparable, or $c \preceq_{\sim_{t,r}} a$. Moreover, for all $a \in A$, there is c in C such that $a \preceq_{\sim_{t,r}} c$. Hence $A \subseteq C$. We know that there is $b \in B$ and $c \in C$ such that $b \prec_{\sim_{t,r}} c$. If $c \in A$, then $c \in B$, which would contradict that B is an anti-chain. Hence $c \notin A$ and we get $A \subsetneq C$;
  - if $A \not\subseteq B$ and $B \subsetneq C$, then for all $a \in A$, there is $b \in B$ – hence $b \in C$ – such that $a \preceq_{\sim_{t,r}} b$. Moreover, there is $a \in A$ and $b \in B$ – hence $b \in C$ – , such that $a \prec_{\sim_{t,r}} b$. Hence we get $A \prec^a_{\sim_{t,r}} C$;
  - if $A \not\subseteq B$ and $B \not\subseteq C$ It is clear that condition (i) of the definition holds, for A and C. For condition (ii), we know that there is $a \in A$ and $b \in B$ such that $a \prec_{\sim_{t,r}} b$. Moreover, there is $c \in C$ such that $b \preceq_{\sim_{t,r}} c$. Hence $a \prec_{\sim_{t,r}} c$.
- *it is asymmetric.* It is because it is irreflexive and transitive.

□

Lemma 10. *Let Σ and alphabet, $t, t' \in T_\Sigma$, $u_1, \ldots, u_n \in N_t$ and $v_1, \ldots, v_n \in N_{t'}$ such that: u_i and u_j are incomparable by $\trianglelefteq$, for $i \neq j$, and for all $i \in \{1, \ldots, n\}$, $path_t(u_0, u_i)$ is isomorphic to $path_t(v_0, v_i)$, where u_0 (resp. v_0) is the root of t (resp. t').*
Then for all $t_1, \ldots, t_n \in T_\Sigma$, if $t = t'$, then $t[u_1 \leftarrow t_1] \ldots [u_n \leftarrow t_n] = t'[v_1 \leftarrow t_1] \ldots [v_n \leftarrow t_n]$, where $t[u_1 \leftarrow t_1] \ldots [u_n \leftarrow t_n]$ is the tree t where the subtree at position u_i has been substituted by t_i, for all $i \in \{1, \ldots, n\}$, and similarly for $t'[v_1 \leftarrow t_1] \ldots [v_n \leftarrow t_n]$.

Proof. This is because if $t = t'$, then $t|_{u_i} = t'|_{v_i}$, for all $i \in \{1, \ldots, n\}$. □

Proof (Proof of Fact 1 of the proof of Lemma 6). Suppose 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[10] that there are $w_1, w_2 \in N_t$ such that $w_1 \trianglelefteq v_1$, $w_2 \trianglelefteq v_2$ and $O_r(w_1) =_A O_r(w_2)$ (hence since $=_A \subseteq id_Q$, $O_r(w_1) = O_r(w_2)$).
Moreover, $w_1, w_2 \in N_{t'}$, otherwise v_1 and v_2 would have been removed from t. We consider two cases:

- If there is no node $u' \in [u]$ such that $w_1 \trianglelefteq u' \trianglelefteq v_1$ or $w_2 \trianglelefteq u' \trianglelefteq v_2$, then we have $\text{path}_{t'}(w_1, v_1)$ isomorphic to $\text{path}_{t'}(w_2, v_2)$. Since $O_{r'}(w_1) = O_r(w_1)$ and $O_{r'}(w_2) = O_r(w_2)$, we get $O_{r'}(w_1) =_A O_{r'}(w_2)$ and therefore $w_1 \sim_{t',r'} w_2$, from which we get $v_1 \sim_{t',r'} v_2$.
- If there is $u' \in [u]$ such that $w_1 \trianglelefteq u' \trianglelefteq v_1$. Let $u'' \in N_t$ such that $w_2 \trianglelefteq u'' \trianglelefteq v_2$ and $\text{path}_t(w_1, u)$ is isomorphic to $\text{path}_t(w_2, u')$. Since $w_1 \sim_{t,r} w_2$, we also have $u'' \sim_{t,r} u'$. Hence $u'' \in [u]$. Since we pump in parallel in both u' and u'', we still have $\text{path}_{t'}(w_1, v_1)$ isomorphic to $\text{path}_{t'}(w_2, v_2)$, 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 $w_1 \trianglelefteq v_1$, $w_2 \trianglelefteq v_2$ and $O_{r'}(w_1) =_A O_{r'}(w_2)$. 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 $w_1 \trianglelefteq v_1$ and $w_2 \trianglelefteq v_2$ in t. Suppose that $\text{path}_t(w_1, v_1)$ is not isomorphic to $\text{path}_t(w_2, v_2)$. We show a contradiction (hence this will prove $v_1 \sim_{t,r} v_2$). Since $\text{path}_t(w_1, v_1)$ and $\text{path}_t(w_2, v_2)$ are not isomorphic, and after pumping $\text{path}_{t'}(w_1, v_1)$ and $\text{path}_{t'}(w_2, v_2)$ are isomorphic, necessarily a pumping has occurred in an elementary context rooted at some node $u' \in [u]$ such that $w_1 \trianglelefteq u' \trianglelefteq v_1$ or $w_2 \trianglelefteq u' \trianglelefteq v_2$. Suppose that $w_1 \trianglelefteq u'$, and let u'' such that $\text{path}_t(w_1, u')$ is isomorphic to $\text{path}_t(w_2, u'')$ (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 $\text{path}_{t'}(w_1, v_1)$ non-isomorphic to $\text{path}_{t'}(w_2, v_2)$, since we pump below u' and u''. Hence u'' belongs to $\text{path}_t(w_2, v_2)$. It is not difficult to show that since we pump in parallel at equivalent positions, then we still have $\text{path}_{t'}(w_1, v_1)$ non-isomorphic to $\text{path}_{t'}(w_2, v_2)$, which is a contradiction.
End of Proof of Fact 1. □

D Applications

D.1 Proof of Theorem 5

The following proposition proves Theorem 5:

[10] 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 $\leftrightarrow_{t,r}$. If there is v_3 such that $v_1 \sim_{t,r} v_3$ and $v_3 \leftrightarrow_{t,r} v_2$, and nodes w_1, w_3 above v_3 such that $O_r(w_3) =_A O_r(w_1)$. By definition of $\leftrightarrow_{t,r}$, there are w_3' and w_2' such that $\text{path}_t(w_3', v_3)$ is isomorphic to $\text{path}_t(w_2', v_2)$, and $O_r(w_3') =_A O_r(w_2')$. Since $=_A \subseteq id_Q$ and we put the same run below states q such that $q =_A q$, we have $r|_{w_3'} = r|_{w_2'}$, and $r|_{w_1} = r|_{w_3}$. Suppose that $w_3 \trianglelefteq w_3'$, and let w_1' above v_1 such that $\text{path}_t(w_1, w_1')$ is isomorphic to $\text{path}_t(w_3, w_3')$: it exists since $r|_{w_1} = r|_{w_3}$, moreover, $O_r(w_1') =_A O_r(w_3')$. Since $O_r(w_3') =_A O_r(w_2')$ and $=_A \subseteq id_Q$, we also have $O_r(w_1') =_A O_r(w_2')$.
The case $w_3' \triangleleft w_3$ is proved similarly.

Proposition 10. *For any closed formula ϕ in $MSO^{\exists}_{=}$, one can compute a vb-TAGED, whose size is non-elementary in the size of ϕ, accepting the models of ϕ. Conversely, for any vbTAGED (A, k), one can compute a closed $MSO^{\exists}_{=}$ formula ϕ whose models are the trees accepted by A (modulo an exponential blow-up).*

Proof. **Forth direction**
First, since we can express in MSO that a set is a singleton, we allow to use first-order variables in predicates eq and diff_k, $k \in \mathbb{N}$.
Now, note that $\neg\mathsf{eq}(X)$ is equivalent to $\exists x \in X \exists y \in X, \mathsf{diff}_1(x, y)$. The formula $\neg\mathsf{diff}_k(X, Y)$ holds in a tree t under assignment ρ if either there is a descendant chain of length strictly greater than k in $\rho(X)$ or $\rho(Y)$ (this is expressible in MSO), or there are two nodes $u \in \rho(X)$ and $v \in \rho(Y)$ such that $t|_u = t|_v$ (which can be expressed by $\exists x \in X \exists y \in Y, \mathsf{eq}(\{x, y\})$).
Hence, every $MSO^{\exists}_{=}$ formula ϕ is equivalent to a disjunction of formulas of the form:

$$\Phi = \exists \overline{X}, \psi(\overline{X}) \wedge \psi_{test}(\overline{X})$$

where $\overline{X}$ is a tuple of set variables, the formula ψ is an MSO-formula, and ψ_{test} is a conjunction of atoms $\mathsf{eq}(X_i)$ or $\mathsf{diff}_k(X_i, X_j)$, where $X_i, X_j \in \overline{X}$. We can easily adapt the proof of the closure by union of TAGED to vbTAGED, so that we only need to prove the proposition for formulas Φ.
We let $\overline{X}$ being equal to $X_1, \ldots, X_n$. We use the classical Thatcher and Wright's construction to transform $\psi(\overline{X})$ into a tree automaton $A = (\Sigma \times \{0, 1\}^n, Q, F, \Delta)$, such that the i-th component of the tuple of any label corresponds to the i-th variable in $\overline{X}$, namely X_i. Then, projecting A on its first component results in a tree automaton recognizing the models of $\exists \overline{X},\ \psi(\overline{X})$. But, instead of projecting the automata as usual, we project the Booleans from the labels into the states like in [22]. We denote by $\mathsf{proj}(A) = (\Sigma, Q_{\mathsf{proj}(A)}, F_{\mathsf{proj}(A)}, \Delta_{\mathsf{proj}(A)})$ the resulting automaton. It is defined by: $Q_{\mathsf{proj}(A)} \subseteq Q \times \{0, 1\}^n$, $F_{\mathsf{proj}(A)} = F \times \{0, 1\}^n$ and $\Delta_{\mathsf{proj}(A)}$ is the set of rules $a((q_1, \overline{b}_1), (q_2, \overline{b}_2)) \to (q, \overline{b})$, such that $\overline{b}_1, \overline{b}_2$ are Boolean tuples, and $(a, \overline{b})(q_1, q_2) \to q$.
Finally, we define the relations $=_{\mathsf{proj}(A)}$ and $\neq_{\mathsf{proj}(A)}$ on states of $\mathsf{proj}(A)$. For all $\overline{b} \in \{0, 1\}^n$, we denote by $\overline{b}(i)$ its i-th projection. Now, $\forall (q_1, \overline{b}_1), (q_2, \overline{b}_2) \in Q_{\mathsf{proj}(A)}$, we let:

- $(q_1, \overline{b}_1) =_{\mathsf{proj}(A)} (q_2, \overline{b}_2)$ if $\exists i \in \{1, \ldots, n\}$ st $\overline{b}_1(i) = \overline{b}_2(i) = 1$ and $\mathsf{eq}(X_i)$ is an atom of ψ_{test};

- $(q_1, \overline{b}_1) \neq_{\mathsf{proj}(A)} (q_2, \overline{b}_2)$ if $(q_1, \overline{b}_1) \neq (q_2, \overline{b}_2)$ and $\exists i, j \in \{1, \ldots, n\}$ st $\overline{b}_1(i) = 1$ and $\overline{b}_2(j) = 1$ and $\mathsf{diff}_k(X_i, X_j)$ is an atom of ψ_{test}, for some k;

Finally, the vbTAGED equivalent to Φ is defined by $\mathsf{proj}(A)$ equipped with $=_{\mathsf{proj}(A)}$ and $\neq_{\mathsf{proj}(A)}$, with the bound $max\{k \mid \exists X \exists Y,\ \mathsf{diff}_k(X, Y)$ is an atom of $\psi_{test}\}$.
It is a bit tedious, but not difficult, to prove that $\forall t \in T_\Sigma, t \in L(A)$ iff $t \models \Phi$.

Complexity The classical Thatcher and Wright's construction which transform an MSO-formula into an equivalent tree automata is known to be non-elementary

in time complexity (in the size of the formula). The size of the tree automata might also be non-elementary in the size of the input formula. Since our TAGED construction relies on this construction, the output TAGED has also a non-elementary size in the size of the input $MSO^{\exists}_{=}$-formula.

Back direction Conversely, let $A = (\Sigma, Q, F, \Delta, =_A, \neq_A)$ be a TAGED, and $k \in \mathbb{N}$. We now consider the vbTAGED (A, k). It is already known (see [7]) that the tree automaton (Σ, Q, F, Δ) is equivalent to an MSO-formula of the form:

$$\phi \;=\; \exists X_{q_1} \ldots \exists X_{q_n} \;\; \phi_\Delta(X_{q_1}, \ldots, X_{q_n})$$

where $\{q_1, \ldots, q_n\} = Q$. Set variables X_{q_i}'s are intended to capture the set of nodes labeled by state q_i in a successful run of A (hence the set denoted by $\{X_{q_1}, \ldots, X_{q_n}\}$ forms a partition of the set of nodes (with possibly empty blocks). Formula $\phi_\Delta(X_{q_1}, \ldots, X_{q_n})$ describes the behavior of the tree automaton, in terms of runs. We do not make this formula explicit and refer the reader to [7] for more details.
The constraints are added by taking ϕ in conjunction with $\bigwedge_{q \neq_A p} \mathsf{diff}_k(X_q, X_p) \wedge \bigwedge_{q =_A p} \mathsf{eq}(X_q \cup X_p)$ (set union is easily expressible in MSO).
Finally, we must ensure that there is not more than k inequality states along paths, with the following formula:

$$\neg \exists x_1 \ldots \exists x_{k+1} \bigwedge_{i=1}^{k} x_i \lhd x_{i+1} \;\; \wedge \bigwedge_{i=1}^{k+1} \bigvee_{q \in \mathrm{dom}(\neq_A)} x_i \in X_q$$

where $\lhd$ denotes the descendant relation, and is definable in MSO.

Finally, we define a formula ϕ_A by:

$$\phi_A \;=\; \begin{cases} \exists X_{q_1} \ldots \exists X_{q_n} & \phi_\Delta(X_{q_1}, \ldots, X_{q_n}) \;\; \wedge \\ & \bigwedge_{q \neq_A p} \mathsf{diff}_k(X_q, X_p) \;\wedge\; \bigwedge_{q =_A p} \mathsf{eq}(X_q \cup X_p) \wedge \\ & \neg \exists x_1 \ldots \exists x_{k+1} \bigwedge_{i=1}^{k} x_i \lhd x_{i+1} \;\wedge \\ & \quad \bigwedge_{i=1}^{k+1} \bigvee_{q \in \mathrm{dom}(\neq_A)} x_i \in X_q \end{cases}$$

By construction, the models of ϕ_A are the trees of $L(A, k)$, and conversely. While formula ϕ_Δ is polynomial in the size of A, the formula ϕ_A is exponential in $log_2(k)$. □

D.2 Unification

Sketch of Proof of Proposition 6 We introduce states for each subterm of t, and a special state $q_\forall$ in which every ground term evaluates: i.e. we add the rules $a \to q_\forall$, and $f(q_\forall, q_\forall) \to q_\forall$, for all $a, f \in \Sigma$. If $x \in \mathcal{X}_t$ occurs in t, we add the ϵ-transition $q_\forall \to q_x$. If $t' = f(t_1, t_2)$ is a subterm of t, we add the rule $f(q_{t_1}, q_{t_2}) \to q_t$, and if $t' = X(t_1, \ldots, t_n)$ is a subterm of t, we add the ϵ-transitions $q_{t_i} \to q_\forall$, $i = 1, \ldots, n$, and $q_\forall \to q_{t'}$. In addition, we add a regular control to check that all states q_{t_i} occurs in the runs, and occurs below $q_{t'}$. Finally, we add the equality constraints $q_x =_A q_x$, for all x occurring in t. □

Proof of Proposition 7 The proof goes similarly as the proof of Proposition 5. We reduce PCP and encode solutions of PCP as in Figure 3. See the beginning of proof of Proposition 5 for details about the notations.
We now construct an equational formula $\phi(x)$ in one free variable x, such that $\phi(x)$ is unsatisfiable iff $\mathcal{I}$ has no solution.
Informally, we first have to check if the tree denoted by x has the shape of Figure 3. This can easily be done by a regular constraint $\phi_{shape} = x \in L_x$. If some tree t satisfies ϕ_{shape}, then all its leaves have to be labeled c, the ternary nodes have to be labeled f, and the unary node have to be labeled in Σ.
Then, for each pattern of the form $f(x_1, f(x_2, x_3, x_4), x_5)$ occurring in the tree, we have to verify that x_1 is of the form $u_i(x_2)$ and x_5 is of the form $v_i(x_4)$ for some i. It is easy to write an equational formula $\phi_{pattern}(x_1, x_2, x_3, x_4, x_5, x)$ which checks whether the nodes bounded to $x_1, \ldots, x_5$ form a pattern $f(x_1, f(x_2, x_3, x_4), x_5)$ in the tree denoted by x. Indeed, we take:

$$\phi_{pattern}(x_1, x_2, x_3, x_4, x_5, x) = \exists X \ X(f(x_1, f(x_2, x_3, x_4), x_5)) = x$$

For every $i \in \{1, \ldots, m\}$, we define a formula $\phi_{u_i}(y, y')$ which checks whether the tree y' is below y and the sequence of labels along the path from the root of y down to the root of y' is u_i. We define $\phi_{v_i}(x, y)$ similarly.

$$\phi_{u_i}(y, y') = \exists X_{u_i}, \ y = X_{u_i}(y') \wedge X_{u_i} \in L_{u_i}$$

where L_{u_i} is the (regular) set of consisting of one unary context C (over unary symbols) such that the path from its root down to the hole is labeled u_i.
Finally, we define $\phi_{init}(x) = \exists x_1 \exists x_2, \ f(x_1, x_2, x_1) = x$ as the formula which checks whether the tree rooted at the first child of x is isomorphic to the tree rooted at the third child of x.
Now, there is a substitution $\sigma : x \mapsto t$ satisfying the following formula iff t is a tree representation of a solution of $\mathcal{I}$:

$$\begin{array}{l} \phi_{shape}(x) \ \wedge \ \phi_{init}(x) \ \wedge \\ \forall x_1 \ldots \forall x_5, \ \phi_{pattern}(x_1, \ldots, x_5, x) \implies \\ \exists y \exists y' \ (y = x_2 \wedge y' = x_4 \wedge \bigvee_{i=1}^{m} \phi_{u_i}(x_1, y) \wedge \phi_{v_i}(x_5, y')) \end{array}$$

□

Proof of Theorem 6 We reduce the problem to deciding if the language of a vbTAGED is empty, which is known to be decidable.
Since formulas are existential, and regular languages are closed by union and intersection, we can assume that ϕ is of the form $\bigvee_i \phi_i$ where each ϕ_i has the following form:

$$\phi_i = \exists x_1 \ldots \exists x_n \exists X_1 \ldots \exists X_m \ \underbrace{\bigwedge_{j=1}^{n'} X_j \in L_{X_j}}_{c_i} \wedge \underbrace{\bigwedge_{k=1}^{n_0} t^1_{0,k} = t^2_{0,k} \wedge \bigwedge_{\ell=1}^{n_1} t^1_{1,\ell} \neq t^2_{1,\ell}}_{e_i}$$

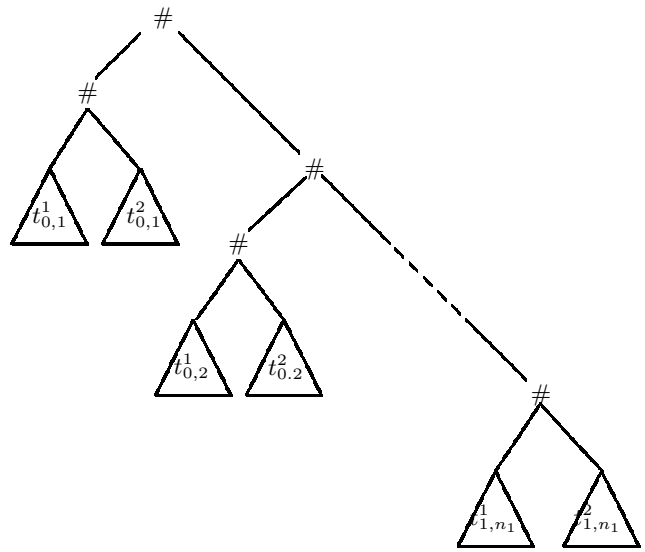

Fig. 8. Term T

where for all context variable X, there is at most one i such that X occurs in $t^1_{0,i}, t^2_{0,i}, t^1_{1,i}$ or $t^2_{1,i}$. Note that there is no membership constraints for term variables, since they can easily translated to constraints of nullary context variables. Testing satisfiability of ϕ is reduced to testing satisfiability of ϕ_i, for all i.

Let $\# \notin \Sigma$ be a fresh symbol. We let T be the following (binary) term (see also Fig 8:

$$T = \#(\#(\underbrace{t^1_{0,1}, t^2_{0,1}}_{=}), \#(, \dots, \#(\#(\underbrace{t^1_{0,n_0}, t^2_{0,n_0}}_{=}), \#(\#(\underbrace{t^1_{1,1}, t^2_{1,1}}_{\neq}), \#(\dots, \#(\underbrace{t^1_{1,n_1}, t^2_{1,n_1}}_{\neq})), \dots)$$

We can extend the proof of Proposition 6 to manage membership constraints, in order to define a vbTAGED A_i such that:

$$L(A_i) = \{T\sigma \mid \sigma \text{ is a ground substitution st } \begin{cases} \sigma(X_j) \in L_{X_j},\ j = 1, \dots, n' \\ t^1_{0,k}\sigma = t^2_{0,k}\sigma,\ k = 1, \dots, n_0 \\ t^1_{1,\ell}\sigma \neq t^2_{1,\ell}\sigma,\ \ell = 1, \dots, n_1 \end{cases} \}$$

By definition of A_i, ϕ_i is satisfiable iff $L(A_i) \neq \varnothing$, which is decidable by Theorem 4. $\square$

E Addional References

References

[22] Joachim Niehren, Laurent Planque, Jean-Marc Talbot and Sophie Tison. N-ary Queries by Tree Automata *10th International Symposium on Database Programming Languages, Springer, 2005*