Information Processing Letters 118 (2017) 6–9

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

# The emptiness problem for tree automata with at least one global disequality constraint is NP-hard

P.-C. Héam [a,*], V. Hugot [b], O. Kouchnarenko [a]

[a] FEMTO-ST – CNRS – Univ. Bourgogne Franche-Comte, France
[b] LIFL – INRIA, France

ARTICLE INFO

*Article history:*
Received 3 July 2014
Received in revised form 3 June 2016
Accepted 12 September 2016
Available online 14 September 2016
Communicated by A. Muscholl

*Keywords:*
Formal languages
Tree automata
Algorithms

ABSTRACT

The model of tree automata with global equality and disequality constraints was introduced in 2007 by Filiot, Talbot and Tison, and extended in various ways since then. In this paper we show that if there is at least one disequality constraint, the emptiness problem is NP-hard.



## 1. Introduction

Tree automata are a pervasive tool of contemporary computer science, with applications running the gamut from XML processing [12] to program verification [4,13,11]. Since their original introduction, they have spawned an ever-growing family of variants, each with its own characteristics of expressiveness and decision complexity. Among them is the family of tree automata with *equality and disequality constraints*, providing several means for comparing subtrees. Examples of such automata are the original class introduced in [14], their restriction to constraints between *brothers* [3], and *visibly tree automata* with memory and constraints [6]. In this paper we focus on a recently introduced variant: tree automata with *global* equality and disequality constraints [8,9], later extended [1,2]. For this class of automata, the universality problem is undecidable [9], while membership is NP-complete [9], and emptiness is decidable [1,2]. Several complexity results for subclasses were pointed out in the literature: the membership problem remains NP-complete for rigid tree automata [13] but it is polynomial for tree automata with a fixed number of equality constraints and no disequality constraints [11]. The emptiness problem is EXPTIME-complete if there are only equality constraints [9], in NEXPTIME if there are only irreflexive disequality constraints [9], and in 3-EXPTIME if there are only reflexive disequality constraints [7]. The latter, closely related to key constraints for XML documents, disallowed in [8,9], were introduced in [1] – we use such constraints in this paper. However, emptiness is decidable in polynomial-time for rigid tree automata [13].

It is known that the emptiness problem is NP-hard for tree automata with global disequality constraints, by reduction of emptiness for DAG[1] automata [5]: cf. [15, Thm. 4.1]. Those automata run on DAG representations of terms, such that identical subterms must be rooted in the same node of the DAG. Therefore any subterms evaluated in different states must be rooted at different positions in the DAG, and thus must be different. The reduction from DAG automata to tree automata with global disequality constraints is easy: the rules are unchanged, and it suffices to add disequality constraints between every couple of distinct states.

* Corresponding author.
*E-mail address:* pheam@femto-st.fr (P.-C. Héam).

[1] Directed acyclic ordered graphs with maximal sharing property.

http://dx.doi.org/10.1016/j.ipl.2016.09.007
0020-0190/

Whereas that reduction requires an arbitrary number of disequality constraints, in this paper we show that a single (reflexive) disequality constraint is sufficient: the emptiness problem is NP-hard for tree automata with global equality and disequality constraints if there is at least one disequality constraint.

## 2. Formal background

A *ranked alphabet* is a finite set $\mathcal{F}$ of symbols equipped with an arity function arity from $\mathcal{F}$ into $\mathbb{N}$. Symbols of arity 0 are called *constants*. The set of *terms* on $\mathcal{F}$, denoted $\mathcal{T}(\mathcal{F})$ is inductively defined as the smallest set satisfying: for every $t \in \mathcal{F}$ such that $\text{arity}(t) = 0$, $t \in \mathcal{T}(\mathcal{F})$, if $t_1, \ldots, t_n$ are in $\mathcal{T}(\mathcal{F})$ and if $f \in \mathcal{F}$ has arity $n > 0$, then $f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F})$. The set of *positions* of a term $t$, denoted $\text{Pos}(t)$, is the subset of $\mathbb{N}^*$ (finite words over $\mathbb{N}$) inductively defined by: if $\text{arity}(t) = 0$, then $\text{Pos}(t) = \{\varepsilon\}$; if $t = f(t_1, \ldots, t_n)$, where $n > 0$ is the arity of $f$, then $\text{Pos}(t) = \{\varepsilon\} \cup \{i \cdot \alpha_i \mid \alpha_i \in \text{Pos}(t_i)\}$, where $\cdot$ denotes the concatenation of positions. A term $t$ induces a function (also denoted $t$) from $\text{Pos}(t)$ into $\mathcal{F}$, where $t(\alpha)$ is the symbol of $\mathcal{F}$ occurring in $t$ at the position $\alpha$. The subterm of a term $t$ at position $\alpha \in \text{Pos}(t)$ is the term $t|_\alpha$ such that $\text{Pos}(t|_\alpha) = \{\beta \mid \alpha \cdot \beta \in \text{Pos}(t)\}$ and for all $\beta \in \text{Pos}(t|_\alpha)$, $t|_\alpha(\beta) = t(\alpha \cdot \beta)$. For any pair of terms $t$ and $t'$, any $\alpha \in \text{Pos}(t)$, the term $t[t']_\alpha$ is the term obtained by substituting in $t$ the subterm rooted at position $\alpha$ by $t'$. Let $\mathcal{X}$ be an infinite countable set of variables such that $\mathcal{X} \cap \mathcal{F} = \emptyset$. A *context* $C$ is a term in $\mathcal{T}(\mathcal{F} \cup \mathcal{X})$ (variables are constants) where each variable occurs at most once; it is denoted $C[X_1, \ldots, X_n]$ if the occurring variables are $X_1, \ldots, X_n$. If $t_1, \ldots, t_n$ are in $\mathcal{T}(\mathcal{F})$, $C[t_1, \ldots, t_n]$ is the term obtained from $C$ by substituting each $X_i$ by $t_i$. The *depth* of a term $t$ is the maximal length of the words in $\text{Pos}(t)$.

A *tree automaton* on a ranked alphabet $\mathcal{F}$ is a tuple $\mathcal{A} = (Q, \Delta, F)$, where $Q$ is a finite set of states, $F \subseteq Q$ is the set of final states and $\Delta$ is a finite set of rules of the form $f(q_1, \ldots, q_n) \to q$, where $f \in \mathcal{F}$ has arity $n$ and the $q_i$'s and $q$ are in $Q$. A tree automaton $\mathcal{A} = (Q, \Delta, F)$ induces a relation on $\mathcal{T}(\mathcal{F} \cup Q)$ (where elements of $Q$ are constant), denoted $\to_\mathcal{A}$ or just $\to$, defined by $t \to_\mathcal{A} t'$ if there exists a transition $f(q_1, \ldots, q_n) \to q \in \Delta$ and $\alpha \in \text{Pos}(t)$ such that $t' = t[q]_\alpha$, $t(\alpha) = f$ and for every $1 \le i \le n$, $t(\alpha \cdot i) = q_i$. The reflexive transitive closure of $\to_\mathcal{A}$ is denoted $\to_\mathcal{A}^*$. A term $t \in \mathcal{T}(\mathcal{F})$ is *accepted* by $\mathcal{A}$ if there exists $q \in F$, such that $t \to_\mathcal{A}^* q$. A *run* $\rho$ in $\mathcal{A}$ for a term $t \in \mathcal{T}(\mathcal{F})$ is a function from $\text{Pos}(t)$ into $Q$ such that if $\alpha \in \text{Pos}(t)$ and $t(\alpha)$ has arity $n$, then $t(\alpha)(\rho(\alpha \cdot 1), \ldots, \rho(\alpha \cdot n)) \to \rho(\alpha)$ is in $\Delta$. An *accepting run* is a run satisfying $\rho(\varepsilon) \in F$. It can be checked that a term $t$ is accepted by $\mathcal{A}$ iff there exists an accepting run $\rho$ for $t$ and, more generally, that $t \to_\mathcal{A}^* q$ if there exists a run $\rho$ for $t$ in $\mathcal{A}$ such that $\rho(\varepsilon) = q$. The set of the terms accepted by $\mathcal{A}$ is denoted $L(\mathcal{A})$.

A tree automaton with global equality and disequality constraints (TAG^ for short, following the notations of [2]) on a ranked alphabet $\mathcal{F}$ is a tuple $(\mathcal{A}, R_1, R_2)$, where $\mathcal{A} = (Q, \Delta, F)$ is a tree automaton on $\mathcal{F}$ and $R_1, R_2$ are binary relations over $Q$. The relation $R_1$ is called the set of equality constraints and the relation $R_2$ the set of disequality constraints. A term $t$ is accepted by $(\mathcal{A}, R_1, R_2)$ if there exists an accepting run $\rho$ for $t$ in $\mathcal{A}$ such that: if $(\rho(\alpha), \rho(\beta)) \in R_1$, then $t|_\alpha = t|_\beta$, and if $\alpha \neq \beta$ and $(\rho(\alpha), \rho(\beta)) \in R_2$, then $t|_\alpha \neq t|_\beta$. The set of the terms accepted by $(\mathcal{A}, R_1, R_2)$ is denoted $L((\mathcal{A}, R_1, R_2))$.

For a ranked alphabet $\mathcal{F}$, let TAG^$(k',k)$ denote the class $(\mathcal{A}, R_1, R_2)$ of TAG^, where $\mathcal{A}$ is a tree automaton over $\mathcal{F}$, $|R_1| \le k'$ and $|R_2| \le k$.

## 3. TAG^ and the Hamiltonian path problem

The paper focuses on proving the following theorem.

**Theorem 1.** *The emptiness problem for* TAG^*(0,1) is NP-hard.*

The proof of Theorem 1 is a reduction from the Hamiltonian Path Problem defined below.

> **Hamiltonian Path Problem**
> **Input:** a directed finite graph $G = (V, E)$, with $|V| \ge 1$;
> **Output:** 1 if there exists a path in $G$ visiting each element of $V$ exactly once, 0 otherwise.

The Hamiltonian Path Problem is known to be NP-complete [10]. A path in a non-empty directed graph visiting each vertex exactly once is called a *Hamiltonian path*. Before proving Theorem 1, let us mention the following direct important consequence, which is the main result of the paper.

**Corollary 2.** *For every fixed $k \ge 1$, and every fixed $k' \ge 0$, the emptiness problem for* TAG^*$(k',k)$ is NP-hard.*

We have divided the proof of Theorem 1 into a sequence of lemmas, that can be sketched as follows. Firstly, we show that in a directed graph $G$ with $n$ vertices (with $n \ge 1$), the number $m_G$ of paths of length $n - 1$ can be computed in time polynomial in $n$ (Lemma 4). Secondly, we show how to construct in time polynomial in $\log(m_G)$ a tree automaton $\mathcal{A}_{m_G}$ accepting a single term having exactly $m_G$ leaves (Lemma 5). Next, we build an automaton accepting encodings of paths of length $n - 1$ in the graph that are not Hamiltonian paths: at least one vertex is visited twice (Lemma 7). Combining these two constructions, one obtains a tree automaton that accepts terms encoding the multisets of cardinality $m_G$ whose elements are encodings of non-Hamiltonian paths of $G$ of length $n - 1$. Adding a global disequality constraint allows us to obtain a TAG^ that accepts terms encoding the sets (rather than multisets) of cardinality $m_G$ whose elements are encodings of non-Hamiltonian paths of $G$ of length $n - 1$. By a direct cardinality argument, this TAG^ accepts at least one term iff there is no Hamiltonian path in $G$ (Lemma 8).

Lemma 3, below, is immediately obtained by a cardinality argument.

**Lemma 3.** *In a directed graph $G$ with $n$ vertices, with $n \ge 1$, there exists a Hamiltonian path iff there is a path of length $n - 1$ that does not visit the same vertex twice.*

**Lemma 4.** *Let $G = (V, E)$ be a non-empty directed graph. One can compute $m_G$ in time polynomial in the size of $G$.*

**Proof.** Let us denote by $m_{G,k,u,v}$, for any $k \geq 0$, any $u \in V$ and any $v \in V$, the number of paths of length $k$ from $u$ to $v$ in $G$. One has $m_{G,k+1,u,v} = \sum_{(u,u') \in E} m_{G,k,u',v}$, and $m_{G,0,u,v} = 1$ if $u = v$ and $m_{G,0,u,v} = 0$ otherwise. Therefore, every $m_{G,k,u,v}$, for $k < |V|$, can be computed recursively in time polynomial in $|V|$. Note that $m_G = \sum_{u,v \in V} m_{G,|V|-1,u,v}$, concluding the proof. □

Note that $m_G \leq |V|^{|V|}$.

Let $\mathcal{F}_1 = \{f, g, A\}$, where $f$ has arity 2, $g$ arity 3, and $A$ is a constant. The following construction aims to build in time polynomial in $\log(m)$ a tree automaton accepting a unique term having exactly $m$ leaves.

Let $m$ be a strictly positive integer and let $\beta_1 \ldots \beta_k$ be the binary representation of $m$ ($\beta_1 = 1$ and $\beta_i \in \{0, 1\}$).

Let $\mathcal{A}_m = (Q_1, \Delta_1, F_1)$ be the tree automaton over $\mathcal{F}_1$, where $Q_1 = \{q_i \mid 1 \leq i \leq k\}$, $F_1 = \{q_k\}$ and $\Delta_1 = \{A \to q_1\} \cup \{f(q_i, q_i) \to q_{i+1} \mid 1 \leq i \leq k-1 \text{ and } \beta_{i+1} = 0\} \cup \{g(q_i, q_i, q_1) \to q_{i+1} \mid 1 \leq i \leq k-1 \text{ and } \beta_{i+1} = 1\}$.

**Lemma 5.** *The tree automaton $\mathcal{A}_m$ can be computed in time polynomial in $k$. Moreover, $L(\mathcal{A}_m)$ is reduced to a single term having exactly $m$ leaves, all labeled by $A$.*

**Proof.** The automaton $\mathcal{A}_m$ has $k$ states and $\Delta_1$ is built directly by reading the $\beta_i$'s. Therefore $\mathcal{A}_m$ can be computed in time polynomial in $k$.

The proof is by induction on $k$. If $k = 1$, then $m = 1 = \beta_1$ (since $m \neq 0$). In this case $Q_1 = F_1 = \{q_1\}$ and $\Delta_1 = \{A \to q_1\}$; therefore $L(\mathcal{A}_1) = \{A\}$ and the lemma result holds.

Now assume that the lemma is true for a fixed $k \geq 1$. Let $2^k \leq m < 2^{k+1}$ and set $m = \beta_1 \ldots \beta_k \beta_{k+1}$, the binary representation of $m$. Two cases may arise:

- $\beta_{k+1} = 0$: In this case, by construction, the terms accepted by $\mathcal{A}_m$ are exactly the terms of the form $f(t_1, t_2)$, with $t_1 \to^*_{\mathcal{A}_m} q_k$ and $t_2 \to^*_{\mathcal{A}_m} q_k$. They correspond to the terms $f(t_1, t_2)$, with $t_1, t_2 \in L(\mathcal{A}_{\frac{m}{2}})$. By induction hypothesis, $L(\mathcal{A}_{\frac{m}{2}})$ is a singleton containing a unique term with $\frac{m}{2}$ leaves, all labeled by $A$. It follows that $L(\mathcal{A}_m)$ accepts a unique term with $2 \cdot \frac{m}{2} = m$ leaves, all labeled by $A$.
- $\beta_{k+1} = 1$: Similarly, the terms accepted by $\mathcal{A}_m$ are exactly the terms of the form $g(t_1, t_2, A)$, with $t_1, t_2 \in L(\mathcal{A}_{\frac{m-1}{2}})$. By induction, it follows that $L(\mathcal{A}_m)$ accepts a unique term with $1 + 2 \cdot \frac{m-1}{2} = m$ leaves, all labeled by $A$.

Therefore, the lemma result holds also for $k + 1$, which concludes the proof. □

Since, by Lemma 4, $m_G$ can be computed in time polynomial in $|V|$, then, using Lemma 5, the construction of $\mathcal{A}_{m_G}$ can be done in time polynomial in $|V|$, proving the following lemma.

**Lemma 6.** *Let $G$ be a non-empty directed graph satisfying $m_G \neq 0$. The tree automaton $\mathcal{A}_{m_G}$ can be computed in polynomial time in the size of $G$.*

The next construction is dedicated to a tree automaton $\mathcal{P}_G^{(2)}$ accepting terms encoding sequences of vertices of $G$ of length $|V|$. More formally, let $G = (V, E)$ be a non-empty directed graph and let $n = |V|$. Let $\mathcal{F}_2 = \{\perp\} \cup \{A_v \mid v \in V\}$, where $\perp$ is a constant and the $A_v$'s are of arity 1. Let $\mathcal{P}_G^{(2)} = (Q_2, \Delta_2, F_2)$ be the tree automaton over $\mathcal{F}_2$, where $Q_2 = \{q_0, \ldots, q_n\}$, $F_2 = \{q_n\}$, and

$$\Delta_2 = \{\perp \to q_0\} \cup \{A_v(q_i) \to q_{i+1} \mid v \in V \text{ and } 0 \leq i \leq n-1\}.$$

The automaton $\mathcal{P}_G^{(2)}$ accepts the set of terms of the form $A_{v_1}(\ldots A_{v_n}(\perp)\ldots)$ of depth $n$ over $\mathcal{F}_2$.

Now let $\mathcal{P}_G^{(3)}$ be the tree automaton $(Q_3, \Delta_3, Q_3 \setminus \{q_\perp\})$ over $\mathcal{F}_2$, with $Q_3 = \{q_\perp\} \cup \{q_v \mid v \in V\}$ and $\Delta_3 = \{\perp \to q_\perp\} \cup \{A_v(q_\perp) \to q_v \mid v \in V\} \cup \{A_v(q_w) \to q_v \mid (w, v) \in E\}$. By construction, the automaton $\mathcal{P}_G^{(3)}$ accepts terms of the form $A_{v_1}(\ldots A_{v_k}(\perp)\ldots)$ where $v_k \ldots v_1$ is a path in $G$ (possibly of length 0).

Let $\mathcal{P}_G^{(4)}$ be the tree automaton $(Q_4, \Delta_4, \{q_{\text{final}}\})$ over $\mathcal{F}_2$, where $Q_4 = \{q_{\text{all}}, q_{\text{final}}\} \cup \{q_v \mid v \in V\}$ and $\Delta_4 = \{\perp \to q_{\text{all}}\} \cup \{A_v(q_{\text{all}}) \to q_v, \ A_v(q_{\text{all}}) \to q_{\text{all}} \mid v \in V\} \cup \{A_v(q_v) \to q_{\text{final}}, \ A_w(q_v) \to q_v \mid v, w \in V\} \cup \{A_v(q_{\text{final}}) \to q_{\text{final}} \mid v \in V\}$. The automaton $\mathcal{P}_G^{(4)}$ accepts the terms of the form $A_{v_1}(\ldots A_{v_\ell}(\perp)\ldots)$ of arbitrary depth on $\mathcal{F}_2$ such that at least two $A_{v_i}$'s are equal.

The tree automata $\mathcal{P}_G^{(2)}$, $\mathcal{P}_G^{(3)}$ and $\mathcal{P}_G^{(4)}$ can be constructed in time polynomial in the size of $G$. Therefore, using classical product of automata, one obtains the following result. The uniqueness of the final state can also be obtained in polynomial time using classical $\varepsilon$-transition removal.

**Lemma 7.** *Let $G = (V, E)$ be a non-empty directed graph. One can compute in time polynomial in $|V|$, a tree automaton $\mathcal{P}_G$ on $\mathcal{F}_2$, with a unique final state, and accepting the terms of the form $A_{v_1}(\ldots A_{v_{|V|}}(\perp)\ldots)$ such that $v_{|V|} \ldots v_1$ is a non-Hamiltonian path in $G$ of length $|V| - 1$.*

Let $G = (V, E)$ be a non-empty directed graph satisfying $m_G \neq 0$. Without loss of generality, one can assume that the set of states of $\mathcal{A}_{m_G} = (Q_1, \Delta_1, \{q_k\})$ and $\mathcal{P}_G = (Q, \Delta, \{q_f\})$ are disjoint except that $q_1 = q_f$. We consider the automaton $\mathcal{D}_G = (Q_5, \Delta_5, F_5)$ over $(\mathcal{F}_1 \cup \mathcal{F}_2) \setminus \{A\}$ defined by: $Q_5 = Q \cup Q_1$, $F_5 = \{q_k\}$ and $\Delta_5 = (\Delta \cup \Delta_1) \setminus \{A \to q_1\}$.

**Lemma 8.** *Let $G$ be a non-empty directed graph satisfying $m_G \neq 0$. The* TAG$^\wedge$ $(\mathcal{D}_G, \emptyset, \{(q_1, q_1)\})$ *can be constructed in time polynomial in the size of $G$. Moreover, it accepts the empty language iff there exists a Hamiltonian path in $G$.*

**Proof.** Using Lemma 5, the terms accepted by $\mathcal{D}_G$ are those of the form $C[t_1, \ldots, t_{m_G}]$, where $C[A, \ldots, A]$ is the unique term accepted by $\mathcal{A}_{m_G}$ and each $t_i$ is accepted by $\mathcal{P}_G$. By Lemmas 5 and 7, and the definition of $\mathcal{D}_G$, it follows that the construction can be done in polynomial time with respect to the size of $G$. With the disequality constraint, $(\mathcal{D}_G, \emptyset, \{(q_1, q_1)\})$ accepts an empty language iff $|L(\mathcal{P}_G)| < m_G$. But, by Lemma 7 $|L(\mathcal{P}_G)|$ is exactly the number of non-Hamiltonian paths in $G$ of length $|V| - 1$. Since $m_G$ is the number of paths of length $|V| - 1$ in $G$, using Lemma 3, $L((\mathcal{D}_G, \emptyset, \{(q_1, q_1)\})) = \emptyset$ iff there exists a Hamiltonian path of length $|V| - 1$ in $G$. □

Assume that the Hamiltonian Path Problem restricted to non-empty directed graphs $G$ such that $m_G \neq 0$ can be solved in polynomial time in the size of $G$. Then, given a non-empty directed graph $G$, one can first test (in polynomial time by Lemma 4) whether $m_G \neq 0$. If $m_G = 0$, then there is no Hamiltonian path in $G$. Otherwise, one can test in polynomial time in the size of $G$ whether there is a Hamiltonian path in $G$. Since the Hamiltonian Path Problem is NP-complete [10], the Hamiltonian Path Problem restricted to non-empty directed graphs $G$ such that $m_G \neq 0$ is NP-complete too. Therefore, Theorem 1 is a direct consequence of Lemma 8.

## 4. Conclusion

In this paper we have proved that the emptiness problem for tree automata with global constraints is NP-hard if there is at least one disequality constraint. It is known that the emptiness problem for tree automata with global constraints with only irreflexive disequality constraints is in NEXPTIME [9], and that it is NP-hard – by reduction of emptiness for DAG automata [5]. If there are only reflexive disequality constraints, emptiness is known to be solvable in 3-EXPTIME [7]. The gap between these bounds is large and deserves to be refined.

## Acknowledgement

The authors would like to thank the anonymous referees for helpful comments and suggestions to improve the quality and the readability of the paper.

## References

[1] Luis Barguñó, Carles Creus, Guillem Godoy, Florent Jacquemard, Camille Vacher, The emptiness problem for tree automata with global constraints, in: Logic in Computer Science, LICS 2010, IEEE Computer Society, 2010, pp. 263–272.

[2] Luis Barguñó, Carles Creus, Guillem Godoy, Florent Jacquemard, Camille Vacher, Decidable classes of tree automata mixing local and global constraints modulo flat theories, Log. Methods Comput. Sci. 9 (2) (2013).

[3] Bruno Bogaert, Sophie Tison, Equality and disequality constraints on direct subterms in tree automata, in: 9th Annual Symposium on Theoretical Aspects of Computer Science, STACS 92, in: Lect. Notes Comput. Sci., vol. 577, Springer, 1992, pp. 161–171.

[4] Yohan Boichut, Thomas Genet, Thomas P. Jensen, Luka Le Roux, Rewriting approximations for fast prototyping of static analyzers, in: Term Rewriting and Applications, 18th International Conference, RTA 2007, in: Lect. Notes Comput. Sci., vol. 4533, Springer, 2007, pp. 48–62.

[5] Witold Charatonik, Automata on DAG representations of finite trees, Research report MPI-I-1999-2-001, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, March 1999, Saarbrücken, Germany, p. 66123.

[6] Hubert Comon-Lundh, Florent Jacquemard, Nicolas Perrin, Visibly tree automata with memory and constraints, Log. Methods Comput. Sci. 4 (2) (2008).

[7] Carles Creus, Adrià Gascón, Guillem Godoy, Emptiness and finiteness for tree automata with global reflexive disequality constraints, J. Autom. Reason. 51 (4) (2013) 371–400.

[8] Emmanuel Filiot, Jean-Marc Talbot, Sophie Tison, Tree automata with global constraints, in: Developments in Language Theory, DLT'08, in: Lect. Notes Comput. Sci., vol. 5257, Springer, 2008, pp. 314–326.

[9] Emmanuel Filiot, Jean-Marc Talbot, Sophie Tison, Tree automata with global constraints, Int. J. Found. Comput. Sci. 21 (4) (2010) 571–596.

[10] Michael R. Garey, David S. Johnson, Computers and Intractability, W.H. Freeman and Company, 1979.

[11] Pierre-Cyrille Héam, Vincent Hugot, Olga Kouchnarenko, On positive TAGED with a bounded number of constraints, in: Implementation and Application of Automata, 17th International Conference, CIAA 2012, in: Lect. Notes Comput. Sci., vol. 7381, Springer, 2012, pp. 329–336.

[12] Haruo Hosoya, Foundations of XML Processing: The Tree-Automata Approach, Cambridge University Press, 2010.

[13] Florent Jacquemard, Francis Klay, Camille Vacher, Rigid tree automata and applications, Inf. Comput. 209 (3) (2011) 486–512.

[14] Jocelyne Mongy, Transformations de noyaux reconnaissables d'arbres. Forêts RATEG, PhD thesis, Laboratoire d'Informatique Fondamentale de Lille, Université de Lille 1, 1981.

[15] Camille Vacher, Tree automata with global constraints for the verification of security properties, PhD thesis, ENS Cachan, 2010.