

Checking in Polynomial Time whether or not a Regular Tree Language is Deterministic Top-Down

Sebastian Maneth^a, Helmut Seidl^b

^a*University of Bremen, Faculty of Informatics, Bibliothekstraße 5, Bremen, 28359, , Germany*

^b*Technical University Munich, Institute for Informatics I2, Boltzmannstr. 3, Garching, 85748, , Germany*

Abstract

It is well known that for a given bottom-up tree automaton it can be decided whether or not there exists deterministic top-down tree automaton that recognized the same tree language. Recently it was claimed that such a decision can be carried out in polynomial time (Leupold and Maneth, FCT'2021); but their procedure and corresponding property is wrong. Here we correct this mistake and present a correct property which allows to determine in polynomial time whether or not a given tree language can be recognized by a deterministic top-down tree automaton. Furthermore, our new property is stated for arbitrary deterministic bottom-up tree automata, and not for minimal such automata (as before).

Keywords:

1. Introduction

Deterministic top-down tree (DTD) automata are strictly weaker than deterministic bottom-up tree (DBU) automata. In fact, even certain finite languages cannot be recognized by DTD automata, such as the language consisting of the two trees $f(a, b)$ and $f(b, a)$. Despite this weakness, DTD automata have certain advantages over DBU automata, e.g., they can be evaluated more efficiently. Several characterizations of DTD languages within the DBU languages have been studied in the literature. Viragh [9] introduced the “path-closed” property and constructs a so called “powerset automaton” for the path closure of a DBU language. A similar method is presented by Gécsec and Steinby [2]. Another approach is the “homogeneous closure” of

Nivat and Podelski [8]; also they construct an automaton which has as state set the powerset of the original one. The exact running time of all these procedures had not been analyzed.

In a recent conference publication by Leupold and Maneth [4] a property was presented which supposedly allows to check whether or not the language given by a minimal deterministic bottom-up tree automaton can be recognized by a deterministic top-down tree automaton.

Their property was an attempt to lift the well-known “subtree exchange property” of Nivat and Podelski [8] to minimal deterministic bottom-up tree automata. Essentially the property means, that if certain transitions are present, for instance the transitions

$$\begin{aligned} f(q_1, q_2) &\rightarrow q \\ f(q_2, q_1) &\rightarrow q \end{aligned}$$

then also other transitions *must* be present in the given automaton; in this case the transitions

$$\begin{aligned} f(q_1, q_1) &\rightarrow q \\ f(q_2, q_2) &\rightarrow q. \end{aligned}$$

Unfortunately, the property of Leupold and Maneth is *wrong*: there is a tree automaton which does not satisfy the property, but which *is* recognizable by a deterministic top-down tree automaton! Such a counter example was identified and kindly communicated to the authors by Christof Löding. The idea of the counter example is as follows. We consider trees of this form.

$$\underbrace{g(\cdots g(f(x, y) \cdots))}_{n \text{ times}}$$

with these properties:

- $x, y \in \{a, b\}$ if $n = 0$
- $x = a$ and $y = b$ if $n > 0$ is odd and
- $x = b$ and $y = a$ if $n > 0$ is even.

The corresponding minimal deterministic bottom-up tree automaton has the set of states $\{q_a, q_b, q, p_1, p_2, p, p'\}$. Its set of final states is $\{q, p_1, p_2, p\}$. It

consists of the following set of transitions.

$$\begin{array}{llll}
a & \rightarrow & q_a & g(p_1) \rightarrow p \\
b & \rightarrow & q_b & g(p_2) \rightarrow p' \\
f(q_a, q_a) & \rightarrow & q & g(p) \rightarrow p' \\
f(q_b, q_b) & \rightarrow & q & g(p') \rightarrow p \\
f(q_a, q_b) & \rightarrow & p_1 & \\
f(q_b, q_a) & \rightarrow & p_2 &
\end{array}$$

According to the property of Leupold and Maneth, the two transitions into the state q demand that also the transitions $f(q_a, q_b) \rightarrow q$ and $f(q_b, q_a) \rightarrow q$ be present in the automaton. Since they are not present, their property erroneously flags this language as *not* recognizable by a deterministic top-down tree automaton. However, as the reader may verify, this language *can* be recognized by a deterministic top-down tree automaton. Thus, it is too strict to demand the presence of the above two transitions into the state q , but, it suffices if such transitions exist into *other* states, as long as these states are both final states (because q is a final state). The new property presented in this paper precisely formulates this idea.

Besides being correct, **our new property has another advantage: it is formalized for arbitrary deterministic bottom-up tree automata, i.e., the automata need *not* be minimal.** Using the example of above, the property states that if there are transitions

$$\begin{array}{ll}
f(q_1, q_1) & \rightarrow q \\
f(q_2, q_2) & \rightarrow q' \\
f(q_1, q_2) & \rightarrow q''
\end{array}$$

then there *may not* a input context such that starting from states q and q' the context is accepting, but starting from q'' the context is rejecting. I.e., if there was such a context, then the tree language cannot be recognized by a deterministic top-down tree automaton.

Note that in the context of XML, several notions of deterministic top-down tree automata for *unranked* trees have been studied [3, 5, 7]. For each of these it takes exponential time to decide whether or not a given regular unranked tree language is in the respective class (because regular expressions are used in the definition of the regular unranked tree language). Unranked top-down and bottom-up tree automata were already earlier considered by Cristau, Löding, and Thomas [1]. For their variants they decide (in at least

exponential time) whether a language given by a deterministic bottom-up unranked tree automaton can be recognized by a deterministic top-down unranked tree automaton. Martens, Neven, and Schwentick [6] present a recent survey of results about deterministic ranked and unranked tree languages.

2. Preliminaries

A ranked alphabet Σ is a finite set of symbols, each one of which has associated with it a non-negative integer called its *arity*. We write $\Sigma = \{f^{(2)}, a^{(0)}, b^{(0)}\}$ to denote that symbol f has arity two, and a and b both have arity zero. The subset of symbols of Σ of rank k is denoted by $\Sigma^{(k)}$.

The set T_Σ of (ranked, finite, ordered) *trees over* Σ is the smallest set T of strings such that whenever $k \geq 0$, $t_1, \dots, t_k \in T$, and $f \in \Sigma^{(k)}$, then also $f(t_1, \dots, t_k) \in T$. For trees of the form $a()$ we simply write a .

We fix a special “variable” symbol x which is not part of any ranked alphabet. A *context over* Σ is a tree c over Σ , however, exactly one leaf of c is labeled by the special symbol x . The set of all contexts over Σ is denoted by \mathbb{C}_Σ . For a tree (or a context) $\xi = f(\xi_1, \dots, \xi_k)$ we define its *set of nodes* as $N(\xi) := \{\epsilon\} \cup \{iu \mid i \in \{1, \dots, k\}, u \in N(\xi_i)\}$. Here ϵ denotes the root node.

Definition 1. A *deterministic bottom-up tree automaton (DBA for short)* A is a tuple (Q, Σ, δ, F) where Q is a finite set of states, Σ is a ranked alphabet, δ is the transition function, and $F \subseteq Q$ is the set of final states. The function δ maps every pair (w, f) with $w \in Q^k$, $f \in \Sigma^{(k)}$, and $k \geq 0$ to a state in Q .

For $q \in Q$ we denote by A_q the DBA obtained from A by changing F into the set $\{q\}$. The transition function of a DBA A is naturally extended from symbols to trees in T_Σ : if $\delta^*(t_i) = q_i$ for $i \in [k]$ and $\delta(q_1 \cdots q_k, f) = q$, then $\delta^*(f(t_1, \dots, t_k)) = q$. The language $L(A)$ of A is defined as $\{t \in T_\Sigma \mid \delta^*(t) \in F\}$.

We also use the notion of an A -run ρ : for a tree $t \in T_\Sigma$ it is a mapping ρ from $N(t)$ to Q such that for all nodes $u \in N(t)$, if u is labeled $f \in \Sigma^{(k)}$, $\rho(u) = q$ and $\rho(ui) = q_i$ for $i \in [k]$, then it must hold that $\delta(q_1 \cdots q_k, f) = q$.

Definition 2. A *deterministic top-down tree automaton (DTA for short)* A is a tuple (Q, Σ, q_0, δ) where Q is a finite set of states, Σ is a ranked alphabet, $q_0 \in Q$ is the initial state, and δ is the transition function. The function δ maps every pair (q, f) with $q \in Q$, $f \in \Sigma^{(k)}$, and $k \geq 0$ to an element of Q^k .

For $q \in Q$ we denote by A_q the DTA obtained from A by changing q_0 into the state q . We define the notion of an A -run: for a tree $t \in T_\Sigma$ it is a mapping ρ from $N(t)$ to Q such that for all nodes $u \in N(t)$, if u is labeled $f \in \Sigma^{(k)}$, $\rho(u) = q$ and $\rho(ui) = q_i$ for $i \in [k]$, then it must hold that $\delta(q, f) = q_1 \cdots q_k$.

Definition 3. A regular tree language L fulfills the exchange property if, for every $t \in L$ and every node $u \in N(t)$, if $t[u \leftarrow f(t_1, \dots, t_k)] \in L$ and also $t[u \leftarrow f(s_1, \dots, s_k)] \in L$, then $t[u \leftarrow f(t_1, \dots, t_{i-1}, s_i, t_{i+1}, \dots, t_k)] \in L$ for each $i = 1, \dots, k$.

3. Checking if a Tree Language is Deterministic Top-Down

We say that a tree language L is *deterministic top-down* if there exists a deterministic top-down tree automaton A such that $L(A) = L$.

Let $A = (Q, \Sigma, \delta, F)$ be a deterministic bottom-up tree automaton (DBA). A state $q \in Q$ is *reachable* if there exists a tree $t \in T_\Sigma$ such that $\delta^*(t) = q$. We say that the DBA A is *reduced* if every state q of A is reachable. Note that for a given DBA it is straightforward to construct in polynomial time an equivalent DBA that is reduced. We therefore assume from now on that each given DBA is reduced.

Definition 4. Let $A = (Q, \Sigma, \delta, F)$ be a DBA. The triple $(q, q', q'') \in Q^3$ is called a *conflict* if there is an input symbol $f \in \Sigma$ of rank $k \geq 2$, a context $c \in \mathbb{C}_\Sigma$, states $p_1, \dots, p_k, p'_1, \dots, p'_k \in Q$, and an index j with $1 \leq j \leq k$ such that the following holds:

- (1) $\delta(p_1 \dots p_k, f) = q$,
- (2) $\delta(p'_1 \dots p'_k, f) = q'$,
- (3) $\delta(p_1 \dots p_{j-1} p'_j p_{j+1} \dots p_k, f) = q''$,
- (4) $\delta^*(q, c) \in F$ as well as $\delta^*(q', c) \in F$, but
- (5) $\delta^*(q'', c) \notin F$.

Note that it follows from the fact that A is a *deterministic* bottom-up tree automaton, that $q \neq q''$ and $q' \neq q''$.

Lemma 5. *If a BTA A has conflicts, then $L(A)$ is not deterministic top-down.*

PROOF. Let $A = (Q, \Sigma, \delta, F)$. Since A has a conflict, there are $(q, q', q'') \in Q^3$, $f \in \Sigma^{(k)}$ with $k \geq 2$, $c \in \mathbb{C}_\Sigma$, $j \in [k]$, and $p_1, \dots, p_k, p'_1, \dots, p'_k \in Q$ such that Conditions (1)–(5) of Definition 4 hold. Let $t_1, \dots, t_k, t'_1, \dots, t'_k \in T_\Sigma$ such that, for $i \in [k]$, $\delta^*(t_i) = p_i$ and $\delta^*(t'_i) = p'_i$. From Condition (4) we know that both $c[f(t_1, \dots, t_k)]$ and $c[f(t'_1, \dots, t'_k)]$ are in $L(A)$. Assume that $L(A)$ is deterministic top-down. Then by the subtree exchange property, also $t = c[f(t_1, \dots, t_{j-1}, t'_j, t_{j+1}, \dots, t_k)] \in L(A)$. However, by Conditions (3) and (5), $t \notin L(A)$. Thus, $L(A)$ is not deterministic top-down. \square

Recall that DTA abbreviates “deterministic top-down tree automaton”. For a given DBA A we now define its “associated DTA”. This associated DTA always accepts a superset of $L(A)$. Later we will show that if A has no conflicts (and is “complete”), then in fact the associated DTA accepts precisely $L(A)$. Let $A = (Q, \Sigma, \delta, F)$. The *associated DTA* of A is the DTA $A' = (2^Q, \Sigma, F, \delta')$. Let Q_0 be a non-empty subset of Q . Let $f \in \Sigma^{(k)}$ with $k \geq 1$. Then let

$$\delta'(Q_0, f) = Q_1 \cdots Q_k$$

where the Q_i are defined as follows. Consider all f -transitions of A which have a right-hand side in Q_0 and denote their left-hand sides by

$$f(q_1^1, \dots, q_k^1), \dots, f(q_1^m, \dots, q_k^m).$$

Then $Q_i = \{q_i^1, \dots, q_i^m\}$ for $i \in [k]$. If no such transitions exist then let $Q_i = \emptyset$ for $i \in [k]$. Finally, for $a \in \Sigma^{(0)}$, if there is an a -transition of A which has a right hand side in Q_0 , then let

$$\delta'(Q_0, a) = \varepsilon.$$

Lemma 6. *Let A be a DBA and let A' be the DTA associated to A . Then $L(A) \subseteq L(A')$.*

PROOF. Let $A = (Q, \Sigma, \delta, F)$. The desired inclusion follows from Claim 1: if $t \in T_\Sigma$ is in $L(A)$, then there is an A -run r on t with $q = r(\varepsilon) \in F$. By Claim 1, there is an A' -run on t starting in any state of A' containing q ; since the start state of A' contains q , we obtain that $t \in L(A')$.

Claim 1: Let $t \in T_\Sigma$ and let r be an A -run on t . Let $Q_0 \in 2^Q$ with $r(\varepsilon) \in Q_0$. There exists an A' -run r' on t such that $r'(\varepsilon) = Q_0$ and $r(u) \in r'(u)$ for all $u \in V(t)$.

The claim is proved by induction on the structure of t . Let $t = a \in \Sigma^{(0)}$. If $r(\varepsilon) = q$ then $\delta(a, \varepsilon) = q$; by the definition of A , $\delta'(Q'_0, a) = \varepsilon$ for every $Q'_0 \in 2^Q$ such that $q \in Q'_0$. Hence $\delta'(Q_0, a) = \varepsilon$ which implies the existence of an A' -run r' with $r'(\varepsilon) = Q_0$.

Let $t = f(t_1, \dots, t_k)$ with $k \geq 1$. By the definition of A' , there is a transition $\delta'(Q_0, f) = Q_1 \cdots Q_k$ such that $r(i) \in Q_i$ for $i \in [k]$; this is because A has an f -transition with left-hand side $(f, r(1) \dots r(k))$ and right-hand side $r(\varepsilon) \in Q_0$. Thus, there is an A' -run r' on t with $r'(\varepsilon) = Q_0$. By induction there are A' -runs r'_i on t_i with $r'_i(\varepsilon) = Q_i$ for $i \in [k]$. Thus, r' exists with $r'(iu) = r'_i(u)$ for $i \in [k]$. \square

Example 7. Let $A = (\{q, q_a, q_b, p, p', p_{ab}, p_{ba}\}, \Sigma, \delta, \{q, p, p_{ab}, p_{ba}\})$ be a DBA with the transition function defined as follows:

$$\begin{aligned} \delta(a, \varepsilon) &= q_a, & \delta(b, \varepsilon) &= q_b, \\ \delta(f, q_a q_a) &= q, & \delta(f, q_b q_b) &= q, \\ \delta(f, q_a q_b) &= p_{ab}, & \delta(f, q_b q_a) &= p_{ba}, \\ \delta(g, p_{ab}) &= p, & \delta(g, p_{ba}) &= p', \\ \delta(g, p') &= p, & \delta(g, p) &= p', \end{aligned}$$

We apply the construction described preceding Lemma 6 to A . The set of final states is $\{q, p, p_{ab}, p_{ba}\}$. This set is now the initial state of our DTA A' . We only discuss the states that are reachable from the initial state. Let us consider the input symbol g . We now collect the left-hand sides of all g -transitions of A which have a right-hand side in F : (g, p_{ab}) and (g, p') . Thus, $\{p_{ab}, p'\}$ is a state of A and

$$\delta'(\{q, p, p_{ab}, p_{ba}\}, g) = \{p_1, p'\}$$

is a transition of A' . In the next step, we collect the left-hand sides of all g -transitions of A which have a right-hand side in $\{p_1, p'\}$: (g, p_{ba}) and (g, p) . Thus $\{p_{ba}, p\}$ is a state. We obtain these two g -transitions of A' :

$$\begin{aligned} \delta'(\{p_{ab}, p'\}, g) &= \{p_{ba}, p\} \\ \delta'(\{p_{ba}, p\}, g) &= \{p_{ab}, p'\}. \end{aligned}$$

Next we consider the input symbol f . The left-hand sides of f -transitions of A which have their right-hand side in the final state $\{q, p, p_{ab}, p_{ba}\}$ are $(f, q_a q_a), (f, q_b q_b), (f, q_a q_b)$, and $(f, q_b q_a)$. Thus

$$\delta'(\{q, p, p_{ab}, p_{ba}\}, f) = \{q_a, q_b\}\{q_a, q_b\}$$

is a transition of A' . For the states $\{p_{ab}, p'\}$ and $\{p_{ba}, p\}$ we add the transitions

$$\begin{aligned}\delta'(\{p_{ab}, p'\}, f) &= \{p_a\}, \{q_b\} \\ \delta'(\{p_{ba}, p\}, f) &= \{q_b\}, \{q_a\}\end{aligned}$$

to A' . Finally, for $x \in \{a, b\}$ we add $\delta'(\{q_a, q_b\}, x) = \varepsilon$ and $\delta'(\{q_x\}, x) = \varepsilon$ to A' .

Let $A = (Q, \Sigma, \delta, F)$ be a DBA and let q_{trap} be a fresh symbol not in Q . The *completion* of A is the DBA $A' = (Q \cup \{q_{\text{trap}}\}, \Sigma, \delta \cup \delta', F)$ where $\delta'(w, f) = q_{\text{trap}}$ whenever $\delta(w, f)$ is undefined for $f \in \Sigma^{(k)}$, $k \geq 0$, and $w \in Q^m$. A *completed DBA* is the completion of some DBA A .

Lemma 8. *Let A be a completed DTA that has no conflicts and let A' be the associated DTA of A . Then $L(A') = L(A)$.*

PROOF. From Lemma 6 we already know that $L(A') \supseteq L(A)$. Hence, it remains to show that $L(A') \subseteq L(A)$. This follows from Claim 2 for $Q_0 = F$ (and taking $c = x_1$).

Claim 2: Let $Q_0 \subseteq 2^Q - \{\emptyset\}$ such that there exists a context $c \in \mathbb{C}_\Sigma$ with (1) for all $q \in Q_0$: $\delta^*(q, c) \in F$ and (2) for all $q \in Q - Q_0$: $\delta^*(q, c) \notin F$. Let $t \in T_\Sigma$. If $t \in L(A'_{Q_0})$ then $t \in \bigcup_{q \in Q_0} L(A_q)$.

The claim is proved by induction on the structure of t . Let $t = a \in \Sigma^{(0)}$. If $t \in L(A'_{Q_0})$ then by the definition of A' there exists a $q \in Q_0$ such that $\delta(a, \varepsilon) = q$, i.e., $t \in L(A_q)$.

Let $t = f(t_1, \dots, t_k)$ for $k \geq 1$. Let Q_1, \dots, Q_k such that $\delta'(Q_0, f) = Q_1 \cdots Q_k$. Assume now that $t \in L(A'_{Q_0})$. By the definition of A' this implies that $Q_i \neq \emptyset$ for every $i \in [k]$. We now want to establish that we can apply the induction hypothesis to each t_i with $i \in [k]$. Let $i \in [k]$ and $q_i \in Q_i$. By the definition of A there are q_j with $j \in [k] - \{i\}$ and the transition $\delta(f, q_1 \cdots q_k) \in Q_0$. Let $c_i = c[f(\tilde{t}_1, \dots, \tilde{t}_{i-1}, x_1, \tilde{t}_{i+1}, \dots, \tilde{t}_k)]$, where c is a context as in the claim and each \tilde{t}_j is an arbitrary tree in $L(A_{q_j})$ for $j \in [k] - \{i\}$. Such trees exist because A is reduced. Thus $\delta^*(q_i, c_i) \in F$. Consider any other $q'_i \in Q_i$.

Then there are q'_i such that $\delta(f, q'_1 \cdots q'_k) \in Q_0$. Since there is no conflict, $\delta(f, q_1 \cdots q_{i-1} q'_i q_{i+1} \cdots q_k) = q'$ and $\delta^*(c, q') \in F$. Thus $\delta^*(q'_i, c_i) \in F$. Now consider some $q'_i \in Q - Q_i$. By the definition of A there is no transition $\delta(f, p_1 \cdots p_{i-1} q'_i p_{i+1} \cdots p_k) \in Q_0$ for any $p_1, \dots, p_k \in Q$. Since A is complete, this implies that $\delta(f, p_1 \cdots p_{i-1} q'_i p_{i+1} \cdots p_k) \in Q - Q_0$ for any $p_1, \dots, p_k \in Q$.

Thus, we can apply the induction hypothesis to Q_i and t_i for $i \in [k]$. we obtain that $t_i \in \bigcup_{q \in Q_i} L(A_{q_i})$. Let q_i be the unique state in Q_i such that $\delta^*(t_i) = q_i$. By the definition of A , $\delta(f, q_1 \cdots q_k) \in Q_0$. Thus $t \in \bigcup_{q \in Q_0} L(A_q)$. \square

From Lemmas 5 and 8 we obtain the following theorem.

Theorem 9. *The language $L(A)$ of a completed DBA A is deterministic top-down iff A has no conflicts.*

We call a state q_t a *trap* state of A , if $q_t \notin F$ and for all $f \in \Sigma^{(k)}$ and $q_1, \dots, q_k \in Q$, $\delta(q_1 \cdots q_k, f) = q_t$ whenever $q_i = q_t$ for some $i \in [k]$. Since transitions containing trap states need not explicitly represented, we define the *size* $|A|$ of the automaton A as

$$|A| = |Q| + \sum \{k + 1 \mid \delta(q_1 \cdots q_k, f) \neq q_t\}$$

where we w.l.o.g. assume that all input symbols of A also occur in transitions without trap states. We now show that testing whether or not a given DBA A has conflicts can be achieved in polynomial time.

Theorem 10. *Let $A = (Q, \Sigma, \delta, F)$ be a deterministic bottom-up tree automaton. Let $n = |Q|$, $m = |\delta|$, and let a be the maximal arity of the symbols in Σ . It is decidable in time $O(n^3 m^2 a)$ whether or not $L(A)$ can be recognized by a deterministic top-down automaton.*

PROOF. In order to verify whether or not A has conflicts, we first construct the set T_0 of all triples satisfying the first three properties. This takes time $O(m^2 k)$.

Then we construct the set T of triples of states reachable from T_0 by means of some context. Technically, the set T is inductively defined by

- $T_0 \subseteq T$;

- Assume that $(q, q', q'') \in T$. Then $(q_1, q'_1, q''_1) \in T$ if there is some input symbol $f \in \Sigma$ of arity $k \geq 1$ some index j with $1 \leq j \leq k$ together with states $p_1, \dots, p_{j-1}, p_{j+1}, \dots, p_k \in Q$ such that
 - $\delta(p_1 \dots p_{j-1} q p_{j+1} \dots p_k, f) = q_1$,
 - $\delta(p_1 \dots p_{j-1} q' p_{j+1} \dots p_k, f) = q'_1$, and
 - $\delta(p_1 \dots p_{j-1} q'' p_{j+1} \dots p_k, f) = q''_1$.

Indeed, determining T from T can be done by repeatedly adding new triples. To find the next triple from a given triple (q, q', q'') , at most all transitions in δ without trap state need to be consulted and for each such transition each position i where the state q may occur. Accordingly, the set T can be constructed in time $O(n^3 \cdot m \cdot a)$.

Then A has a conflict iff $T \cap F \times F \times (Q \setminus F) \neq \emptyset$. Given the set T , this check can be implemented in time $O(n^3)$. Thus, altogether the algorithm requires time $O(m^2 \cdot a + n^3 \cdot m \cdot a) \leq O(n^3 \cdot m^2 \cdot a)$. \square

References

- [1] J. Cristau, C. Löding, and W. Thomas. Deterministic automata on unranked trees. In M. Liskiewicz and R. Reischuk, editors, *Fundamentals of Computation Theory, 15th International Symposium, FCT 2005, Lübeck, Germany, August 17-20, 2005, Proceedings*, volume 3623 of *Lecture Notes in Computer Science*, pages 68–79. Springer, 2005.
- [2] F. Gécseg and M. Steinby. *Tree Automata*. Akadémiai Kiadó, Budapest, Hungary, 1984.
- [3] W. Gelade, T. Idziaszek, W. Martens, F. Neven, and J. Paredaens. Simplifying XML schema: Single-type approximations of regular tree languages. *J. Comput. Syst. Sci.*, 79(6):910–936, 2013.
- [4] P. Leupold and S. Maneth. Deciding top-down determinism of regular tree languages. In E. Bampis and A. Pagourtzis, editors, *Fundamentals of Computation Theory - 23rd International Symposium, FCT 2021, Athens, Greece, September 12-15, 2021, Proceedings*, volume 12867 of *Lecture Notes in Computer Science*, pages 341–353. Springer, 2021.
- [5] W. Martens. *Static analysis of XML transformation and schema languages*. PhD thesis, Hasselt University, 2006.

- [6] W. Martens, F. Neven, and T. Schwentick. Deterministic top-down tree automata: past, present, and future. In J. Flum, E. Grädel, and T. Wilke, editors, *Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas]*, volume 2 of *Texts in Logic and Games*, pages 505–530. Amsterdam University Press, 2008.
- [7] W. Martens, F. Neven, T. Schwentick, and G. J. Bex. Expressiveness and complexity of XML schema. *ACM Trans. Database Syst.*, 31(3):770–813, 2006.
- [8] M. Nivat and A. Podelski. Minimal ascending and descending tree automata. *SIAM J. Comput.*, 26(1):39–58, 1997.
- [9] J. Virág. Deterministic ascending tree automata I. *Acta Cybern.*, 5(1):33–42, 1980.