Theoretical Computer Science 454 (2012) 231–239

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Lower bounds for the size of deterministic unranked tree automata

Xiaoxue Piao, Kai Salomaa [*]

School of Computing, Queen's University, Kingston, Ontario K7L 3N6, Canada

ARTICLE INFO

Keywords:
Tree automata
Unranked trees
State complexity
Lower bounds

ABSTRACT

Tree automata operating on unranked trees use regular languages, called horizontal languages, to define the transitions of the vertical states that define the bottom-up computation of the automaton. It is well known that the deterministic tree automaton with smallest total number of states, that is, number of vertical states and number of states used to define the horizontal languages, is not unique and it is hard to establish lower bounds for the total number of states. By relying on existing bounds for the size of unambiguous finite automata, we give a lower bound for the size blow-up of determinizing a nondeterministic unranked tree automaton. The lower bound improves the earlier known lower bound that was based on an ad hoc construction.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

Modern applications of tree automata, such as XML document processing, use unranked trees where the label of a node does not determine the number of children and there is no a priori bound on the number of children of a node [3,14,16,25]. These modern applications have renewed interest also in the descriptional complexity of tree automata [13,18,20], and many problems turn out to be essentially different than the corresponding problems for finite automata operating on strings [28] or for tree automata operating on ranked trees [3,6]. An early reference on regular unranked tree languages is [2].

The set of transitions of an unranked tree automaton is, in general, infinite and the transitions are specified in terms of regular languages, called *horizontal languages.* The bottom-up computation uses a finite number of *vertical states* and to each vertical state q and input symbol σ, is associated a nondeterministic (or deterministic) finite automaton (NFA or DFA) that recognizes the horizontal language of q and σ. The latter are called the horizontal NFAs (or DFAs) of the tree automaton.

The tree automaton is deterministic (a DTA) if for each input symbol σ and two distinct vertical states q_1 and q_2, the horizontal languages associated, respectively, with σ and q_1 and with σ and q_2 are disjoint. We use DTA(DFA) (respectively, NTA(NFA)) to denote the class of deterministic (respectively, nondeterministic) tree automata where the horizontal languages are specified by a DFA (respectively, an NFA). Naturally other variants of the definition can also be considered [3,18,19].

The size of an unranked tree automaton is defined by the number of vertical states and the number of states used by the horizontal automata [3,13,18]. It is known that a regular tree language does not, in general, have a unique minimal DTA(DFA) and minimization of DTA(DFA)s is NP-complete [13]. Since minimization is intractable, lower bounds for this model need to be established by ad hoc techniques, however, no general method comparable to the fooling set techniques used for NFAs is known [1,7,9,11].

By a *state trade-off* we mean a situation where by adding more vertical states to a DTA(DFA) we can reduce the total size of the automaton. The hardness of DTA(DFA) minimization is caused by the existence of state trade-offs [13] and this phenomenon makes it hard to prove lower bounds for the total size of a DTA(DFA) recognizing a given regular tree

[*] Corresponding author. Tel.: +1 613 533 6073; fax: +1 613 533 6513.
E-mail addresses: piao@cs.queensu.ca (X. Piao), ksalomaa@cs.queensu.ca (K. Salomaa).

0304-3975/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2012.03.043

language [18,19]. It should be mentioned that, although the model discussed here is perhaps the commonly used definition of determinism for unranked tree automata [3], there are two alternative definitions of determinism that guarantee the uniqueness of the minimal automaton. These are the syntactically deterministic tree automata of [4,21] and the step-wise tree automata considered in [3,13]. The state complexity of transformations between different unranked tree automaton models has been studied in [13,18].

A nondeterministic unranked tree automaton can be determinized using an extension of the standard subset construction, however, the size blow-up for the horizontal automata, in the general construction, is particularly bad [18]. On the other hand, since we do not have general techniques to establish lower bounds for the size of DTA(DFA)s it is not known what is the precise state complexity of determinization. An ad hoc lower bound construction is given in [18] where converting an NTA(NFA) to a DTA(DFA) causes an exponential blow-up in the number of vertical states and, additionally, the size of each of the exponential number of horizontal DFAs is considerably larger than the original horizontal automaton. However, the lower bound is far from the upper bound and the construction relies on properties of particular types of unary languages which seem too restrictive to handle the general case.

As mentioned above, the difficulty of establishing lower bounds for the size of DTA(DFA)s is caused by state trade-offs where we can reduce the total size of an automaton A by replacing a vertical state q with equivalent copies $q_1, \dots, q_k$ [19]. Here we develop a lower bound criterion that, roughly speaking, from the horizontal languages associated with an input symbol σ and $q_1, \dots, q_k$ constructs an unambiguous finite automaton (UFA) for the horizontal language of A associated with q and σ. Then if A is chosen to be the DTA(DFA) with the minimal number of vertical states (which is unique), we get a lower bound for the total size of horizontal DFAs associated, respectively, with σ and $q_i, i = 1, \dots, k$. However, we need additional assumptions in order to construct an UFA as described above. The limitations of the method are discussed in more detail in the last section of the paper.

As our main result we give an improved lower bound for the state complexity of determinizing a nondeterministic unranked tree automaton. For this purpose, in Section 3, we define a particular family of *comb tree languages*. A tree language L_{comb} of this type is designed in a way that if A is an arbitrary DTA(DFA) recognizing L_{comb}, we can construct from the horizontal DFAs associated with a symbol σ and a set of equivalent vertical states, an UFA for the corresponding horizontal language of the equivalent DTA(DFA) with the minimal number of vertical states. By relying on known lower bounds for the size of UFAs [12], this gives a lower bound for the size of horizontal DFAs of A.

The comb tree languages are defined in terms of two regular string languages L_1 and L_2. In order to optimize the lower bound for the NTA(NFA)-to-DTA(DFA) transformation we can choose L_1 and L_2 to be languages each having a small NFA where L_1 needs a large UFA and L_2 needs a large DFA. The main result will be presented in Section 4.

To conclude the introduction, we recall that Schmidt [24] first established an exponential trade-off between the sizes of NFAs and UFAs. A lower bound 2^{n-1} for the size of an UFA equivalent to an n-state NFA was given in [27] and the lower bound was further refined to $2^n - 1$ in [12]. A comprehensive reference on the descriptional complexity of various restricted NFA models can be found in [8]. Recent work on the state complexity of finite automata on strings includes [5,10,23], and the state complexity of UFAs has been considered in [17].

2. Preliminaries

We assume that the reader is familiar with finite automata operating on strings [22,26,28] and here just fix a few notations. The set of nonnegative integers is denoted $\mathbb{N}$. When there is no confusion, a single element set $\{b\}$ can be denoted as b, for short. The cardinality of a finite set S is denoted $|S|$. The empty string is ε.

A nondeterministic finite automaton (NFA) is a tuple $A = (Q, \Sigma, \delta, I, F)$ where Q is the finite set of states, Σ is the input alphabet, $\delta : Q \times \Sigma \to 2^Q$ is the transition relation and $I \subseteq Q$ (respectively, $F \subseteq Q$) is the set of initial (respectively, final) states. Following [12] we allow an NFA to have multiple initial states. An NFA A is *deterministic* (a DFA) if $|I| = 1$ and for all $q \in Q$ and $\sigma \in \Sigma$, $|\delta(q, \sigma)| \leq 1$. Note that we allow a deterministic automaton to have undefined transitions. An NFA A is *unambiguous* (an UFA) if any string $w \in \Sigma^*$ has at most one accepting computation. The number of states of an NFA A is $|A|$. The state complexity of a regular language L, $\text{sc}(L)$ (respectively, the unambiguous state complexity of L, $\text{usc}(L)$) is the size of the minimal incomplete DFA (respectively, a minimal UFA) recognizing L.

For DFA $A = (Q, \Sigma, \delta, I, F)$ and $\sigma \in \Sigma$ we define the equivalence relation $\sim_{A,\sigma} \subseteq Q \times Q$ by setting

$$q_1 \sim_{A,\sigma} q_2 \quad \text{iff } \delta(q_1, \sigma) = \delta(q_2, \sigma), \quad (q_1, q_2 \in Q).$$

Above $\delta(q_1, \sigma) = \delta(q_2, \sigma)$ allows the possibility that both sides of the equality are undefined. We denote by $\#_\sigma(A)$ the cardinality of the set $\delta(Q, \sigma)$. Note that, if for all states of A the σ-transition is defined, the number of equivalence classes of $\sim_{A,\sigma}$ is equal to $\#_\sigma(A)$, and otherwise it is equal to $\#_\sigma(A) + 1$. For a regular language L, we denote by $\#_\sigma(L)$ the value $\#_\sigma(B_L)$ where B_L is the minimal DFA for L.

2.1. Unranked tree automata

Here we briefly recall some notations used for automata operating on unranked trees. For more details the reader is referred, e.g., to [3,18].

A *tree domain* is a prefix-closed subset D of $\mathbb{N}^*$ such that always when $ui \in D$, $u \in \mathbb{N}^*$, $i \in \mathbb{N}$ and $1 \le j < i$, then also $uj \in D$. The set of nodes of a tree t is represented in the well-known way as a tree domain $\mathrm{dom}(t)$ and when the nodes are labeled by elements of an alphabet Σ the tree t can be viewed as a mapping $\mathrm{dom}(t) \to \Sigma$. We consider unranked trees and hence the node label does not determine the number of children of the node. The set of unranked trees over an alphabet Σ is T_Σ.

For $t_1, t_2 \in T_\Sigma$ and $u \in \mathrm{dom}(t_1)$, $t_1(u \leftarrow t_2)$ is the tree obtained from t_1 by replacing the subtree at node u with t_2. The subtree of t_1 at node u is $(t_1)_u$. For $L \subseteq T_\Sigma$, the Nerode-congruence of L is defined by setting, for $t_1, t_2 \in T_\Sigma$,

$$t_1 \cong_L t_2 \quad \text{iff } [(\forall t \in T_\Sigma)(\forall u \in \mathrm{dom}(t))t(u \leftarrow t_1) \in L \Leftrightarrow t(u \leftarrow t_2) \in L].$$

A *nondeterministic* unranked bottom-up tree automaton (NTA) is a 4-tuple $A = (Q, \Sigma, \delta, F)$ where Q is a finite set of states, Σ is the alphabet, $F \subseteq Q$ is the set of final states, δ is a mapping that associates to each $q \in Q$ and $\sigma \in \Sigma$ a regular language $\delta(q, \sigma) \subseteq Q^*$, called the *horizontal language* of q and σ.

The set of *configurations* of A, $T_\Sigma(Q)$, consists of Σ-trees where some of the leaves may be labeled by states of Q. The single-step computation relation of A, $\vdash_A \subseteq T_\Sigma(Q) \times T_\Sigma(Q)$, is defined by setting $t_1 \vdash_A t_2$ if there exists $u \in \mathrm{dom}(t_1)$ such that $(t_1)_u = \sigma(q_1, \ldots, q_m)$, $\sigma \in \Sigma$, $q_i \in Q$, $i = 1, \ldots, m$, and $t_2 = t_1(u \leftarrow q)$ for some $q \in Q$ such that $q_1 \cdots q_m \in \delta(q, \sigma)$. That is, t_2 is obtained from t_1 by replacing an occurrence of a height-one subtree $\sigma(q_1, \ldots, q_m)$ by q where the sequence of states labeling leaves of the subtree belongs to the horizontal language associated with q and σ.

For $t \in T_\Sigma(Q)$, $t^A \subseteq Q$ denotes the set of states that in some bottom-up computation A may reach at the root of t, that is, $t^A = \{q \in Q \mid t \vdash_A^* q\}$. The set of configurations accepted by A is defined as $L_{\mathrm{config}}(A) = \{t \in T_\Sigma(Q) \mid (\exists q \in F)\ q \in t^A\}$, and the *tree language recognized by* A is $L(A) = L_{\mathrm{config}}(A) \cap T_\Sigma$. The tree language $L(A)$ consists of all Σ-trees t such that some computation of A reaches the root of t in a state of F. Without loss of generality we assume that A has no useless states, that is, for all $q \in Q$ there exist $t_1, t_2 \in T_\Sigma$ such that $q \in t_1^A$ and for some $u \in \mathrm{dom}(t_2)$, $t_2(u \leftarrow q) \in L_{\mathrm{config}}(A)$.

An NTA $A = (Q, \Sigma, \delta, F)$ is said to be *deterministic*, a DTA, if for any two states $q_1, q_2 \in Q$, $q_1 \neq q_2$, and $\sigma \in \Sigma$, we have $\delta(q_1, \sigma) \bigcap \delta(q_2, \sigma) = \emptyset$. If A is deterministic, for any $t \in T_\Sigma$, t^A is a singleton set or empty. As common when dealing with unranked tree automata [19], we allow that some of the horizontal languages of a DTA A may be empty, i.e., A need not be complete.

An NTA(NFA) is a nondeterministic unranked tree automaton where each horizontal language is specified by an NFA. Similarly, a DTA(DFA) is a deterministic tree automaton where each horizontal language is specified by a DFA. The DFAs (respectively NFAs) specifying the horizontal languages are called *horizontal DFAs (respectively NFAs)* and their states are called *horizontal states*. If $A = (Q, \Sigma, \delta, F)$ is an NTA(NFA) (respectively, DTA(DFA)) the horizontal NFA (respectively, DFA) recognizing the language $\delta(q, \sigma)$, $q \in Q$, $\sigma \in \Sigma$, is denoted $H^A_{q,\sigma}$. The states of Q are called *vertical states*.

We define the *(state) size of A*, size(A), as a pair of integers $[|Q|; n]$, where n is the sum of the sizes of all horizontal automata $H^A_{q,\sigma}$ that recognize a nonempty language. If for $q \in Q$, $\sigma \in \Sigma$, the horizontal language $\delta(q, \sigma)$ is empty, the description of A does not include $H^A_{q,\sigma}$ (and we do not add one to the size of A for a one-state automaton recognizing the empty language). When comparing sizes of automata, $[m_1; m_2] \ge [n_1; n_2]$ means $m_i \ge n_i$, $i = 1, 2$.

We say that a DTA(DFA) A is *v-minimal* if A has the smallest number of vertical states among all the equivalent DTA(DFA)s, and A is *t-minimal* if A has the smallest total number of vertical and horizontal states. A regular tree language does not need to have a unique t-minimal DTA(DFA) [13,19].

Using the extension of the Nerode congruence from strings to trees $\cong_L$, we can define the v-minimal DTA(DFA) for a regular tree language L where the states consist of the congruence classes [4], and hence the v-minimal DTA(DFA) is unique (up to isomorphism). As noted in [2–4], for unranked tree languages, in addition to the congruence $\cong_L$ having a finite index we need further conditions to guarantee regularity of the horizontal languages, however, here we just need a v-minimal deterministic automaton for a tree language that is known to be regular, and we do not need to consider the additional conditions. The extension of the Nerode congruence is called the top-congruence in [2], and the corresponding construction for tree languages over ranked alphabets can be found in [6].

Let $A = (Q, \Sigma, \delta, F)$ be an arbitrary DTA(DFA). We say that states $q_1, q_2 \in Q$ are *equivalent*, $q_1 \equiv_A q_2$, if

$$(\forall t \in T_\Sigma)(\forall u \in \mathrm{dom}(t))\ t(u \leftarrow q_1) \in L_{\mathrm{config}}(A) \text{ iff } t(u \leftarrow q_2) \in L_{\mathrm{config}}(A). \tag{1}$$

The condition $q_1 \equiv_A q_2$ means that the states q_1 and q_2 are equivalent in terms of the vertical computation of A, however, the horizontal languages associated with q_1 and q_2 need not coincide. The latter observation gives rise to the possibility of having trade-offs [19] between the numbers of vertical and horizontal states, respectively. Directly from the definition it follows that if A is a DTA(DFA) and $t_1, t_2 \in T_\Sigma$,

$$t_1^A = t_2^A \text{ implies } t_1 \cong_{L(A)} t_2. \tag{2}$$

Based on A we can define a quotient automaton $A/_{\equiv_A}$ whose set of states consists of the $\equiv_A$-classes with the transitions defined in the natural way and show that $A/_{\equiv_A}$ is the v-minimal DTA(DFA) equivalent to A. For establishing our lower bound we need only the above definition of the equivalence relation $\equiv_A$ given in (1).

3. Comb tree languages

We want to relate the total size of the horizontal DFAs of a DTA(DFA) A to the size of UFAs for the horizontal languages of the v-minimal automaton equivalent to A. Results similar to Lemma 3 below could be established under more general assumptions, however, in order to avoid making the notations unnecessarily complicated, we establish the lower bound only for the particular type of comb tree languages that will be used in the proof of our main result in the next section.

Roughly speaking, the *comb tree language* corresponding to string languages L_1 and L_2 consists of all trees where the rightmost branch spells out a string of L_2, and all children of each node of the rightmost branch, except the rightmost child, are leaves and their labels spell out a string of L_1. First we introduce some preliminary notation and after that we define the comb tree languages and establish some technical lemmas for any DTA(DFA) recognizing a comb tree language.

For $t \in T_\Sigma$, the *right branch of* t is defined as

$$\begin{aligned}\mathrm{rb}(t) \ = \{&(u_0, u_1, \ldots, u_k) \mid u_i \in \mathrm{dom}(t), u_{j+1} \text{ is the rightmost child of } u_j,\\ &u_0 = \varepsilon,\ u_k \text{ is a node of height one },\ 0 \le i \le k, 0 \le j \le k-1\}\end{aligned}$$

and the *right-string of* t is $\mathrm{rstring}(t) = t(u_k)t(u_{k-1}) \cdots t(u_0)$ where $(u_0, u_1, \ldots, u_k) = \mathrm{rb}(t)$. The string rstring($t$) consists of the sequence of symbols of Σ labeling bottom-up the rightmost branch of t, and excluding the label of the rightmost leaf of t. With rb(t) as above, the unordered set $\{u_0, u_1, \ldots, u_k\}$ is denoted as RB(t).

For $u \in \mathrm{dom}(t)$ where u is not a leaf, the *child-string of* t *at node* u, $\mathrm{cstring}(t, u) \in \Sigma^*$, is $t(v_1) \cdots t(v_{r-1})$ if $v_1, \ldots, v_r$ are the children of u in left-to-right order. Note that cstring(t, u) does not include the label of the rightmost child of u.

Let $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \{\#_\Sigma\}$ where Σ_1 and Σ_2 are disjoint sets not containing the special symbol $\#_\Sigma$. Let $L_i \subseteq \Sigma_i^*$ be a string language, $i = 1, 2$. In the following of this section, unless otherwise mentioned, Σ is as above and L_1, L_2 denote string languages over Σ_1 and Σ_2, respectively.

The *comb tree language* of L_1 and L_2 is defined as

$$\begin{aligned}\mathrm{comb}_\Sigma(L_1, L_2) \ = \{t \in T_\Sigma \mid \mathrm{rstring}(t) \in L_2,\ (\forall u \in \mathrm{RB}(t))\ \mathrm{cstring}(t, u) \in L_1,\\ \text{and the rightmost leaf of } t \text{ is labeled by } \#_\Sigma\}.\end{aligned} \tag{3}$$

The tree language $\mathrm{comb}_\Sigma(L_1, L_2)$ consists of trees t where the nodes of the rightmost branch of t, from the node of height one to the root, are labeled by a sequence of symbols belonging to L_2, and for each rightmost node u of t that is not a leaf, the string cstring(t, u) is in the language L_1. As a simple example consider the case where $\Sigma_1 = \{a, b\}$, $\Sigma_2 = \{c, d\}$ and $L_1 = \{ab\}$, $L_2 = \{cd\}$. In this case $\mathrm{comb}_\Sigma(L_1, L_2)$ is a singleton tree language consisting of the tree $d(a, b, c(a, b, \#_\Sigma))$.

The following technical property will be used later.

Lemma 1. *For any strings* $\tau_1 \cdots \tau_k \in L_1$, $\rho_1 \cdots \rho_m \in L_1$, $\tau_i, \rho_j \in \Sigma_1$, $1 \le i \le k$, $1 \le j \le m$, $\sigma \in \Sigma_2$ *and* $t \in T_\Sigma$, *we have*

$$\sigma(\tau_1, \ldots, \tau_k, t) \cong_{\mathrm{comb}_\Sigma(L_1,L_2)} \sigma(\rho_1, \ldots, \rho_m, t).$$

Proof. Denote $t_1 = \sigma(\tau_1, \ldots, \tau_k, t)$ and $t_2 = \sigma(\rho_1, \ldots, \rho_m, t)$. The claim follows from the observation that $\mathrm{rstring}(t_1) = \mathrm{rstring}(t_2)$, the strings $\mathrm{cstring}(t_1, \varepsilon)$ and $\mathrm{cstring}(t_2, \varepsilon)$ are both in L_1 and if $u_{i,j}$, $1 \le i \le 2, j \ge 1$, is the jth node of $\mathrm{rb}(t_i)$ below the root, then $\mathrm{cstring}(t_1, u_{1,j}) = \mathrm{cstring}(t_2, u_{2,j})$. □

Suppose that a DTA(DFA) $A = (Q, \Sigma, \delta, F)$ recognizes $\mathrm{comb}_\Sigma(L_1, L_2)$. Denote $\Sigma_1' = \Sigma_1$, $\Sigma_2' = \Sigma_2 \cup \{\#_\Sigma\}$ and for $i = 1, 2$ define

$$Q_i = \{q \in Q \mid (\exists t \in T_\Sigma) : t(\varepsilon) \in \Sigma_i',\ t^A = q\}. \tag{4}$$

The set Q_i consists of states of Q that on some input A may assign to a node labeled by an element of Σ_i'. (Note that Σ_2' contains also the special symbol $\#_\Sigma$.) In the rest of this section, A, Q_1 and Q_2 are always as above.

Suppose that $t_i(\varepsilon) \in \Sigma_i'$, where t_i^A is defined, $i = 1, 2$. Recalling our assumption that A contains no useless states, directly from the definition of $\mathrm{comb}_\Sigma(L_1, L_2)$ it follows that $t_1 \not\cong_{\mathrm{comb}_\Sigma(L_1,L_2)} t_2$ and hence (2) implies that $Q_1 \cap Q_2 = \emptyset$. (Note that if the vertical computation of A contained a useless state, e.g. a dead state, this could belong to both Q_1 and Q_2.)

We want to establish a correspondence between the horizontal languages of A and the language L_1. Since A does not have useless states, for any $\sigma \in \Sigma_2$ and $q \in Q_2$ the horizontal language $\delta(q, \sigma)$ is a subset of $Q_1^* Q_2$. For $p \in Q_2$ and $\sigma \in \Sigma$ we define

$$E(p, \sigma) = \{w \in Q_1^* \mid (\exists q \in Q_2)\ wp \in \delta(q, \sigma)\}. \tag{5}$$

The set $E(p, \sigma)$ consists of all strings w over Q_1 such that in some successful computation of A the sequence of leaves below a node labeled σ can have states labeled by the sequence wp.

For Lemma 2, we need the following technical condition. We say that a language $L \subseteq \Sigma^*$ is Σ*-separating* if each element of Σ occurs in some string of L and no two distinct symbols of Σ are equivalent with respect to the Nerode congruence of L.

Lemma 2. *Assume that* L_1 *is* Σ_1*-separating. Then there exists a bijection* $\varphi : \Sigma_1 \to Q_1$ *such that when* φ *is extended as a morphism* $\Sigma_1^* \to Q_1^*$, *for any* $p \in Q_2$ *and* $\sigma \in \Sigma_2$ *such that* $E(p, \sigma) \neq \emptyset$:

$$\varphi(L_1) = E(p, \sigma).$$

Proof. For $\omega \in \Sigma_1$ we define $\varphi(\omega) = r$ where r is chosen to be the state that A assigns to a leaf labeled by ω, that is, $\varepsilon \in \delta(r, \omega)$. Since ω occurs as a leaf in trees of $\mathrm{comb}_\Sigma(L_1, L_2)$ the state r exists, and since A is deterministic it is unique. Furthermore, since no two distinct symbols $\omega_1, \omega_2 \in \Sigma_1$ are equivalent with respect to the Nerode congruence of L_1, A cannot assign the same state to ω_1 and ω_2, and hence φ is an injective mapping from Σ_1 to Q_1. The mapping φ is surjective because A has no useless states and states of Q_1 can appear only at nodes labeled by elements of Σ_1 (and such nodes are all leaves).

Since $E(p, \sigma) \neq \emptyset$, there exists $q \in Q_2$ such that $\delta(q, \sigma)$ contains a string in $Q_1^* p$. Choose $t_p \in T_\Sigma$ such that $t_p^A = p$, that is, the computation of A assigns state p to the root. Since $\emptyset \neq \delta(q, \sigma) \subseteq Q_1^* Q_2$ and the state q is useful, there exist $\tau_1, \ldots, \tau_k \in \Sigma_1$ such that $\sigma(\tau_1, \ldots, \tau_k, t_p)$ is a subtree of a tree in $\mathrm{comb}_\Sigma(L_1, L_2)$.

Let $w = \gamma_1 \cdots \gamma_m \in L_1$ be arbitrary, $\gamma_i \in \Sigma_1$, $i = 1, \ldots, m$. Now Lemma 1 implies that also $\sigma(\gamma_1, \ldots, \gamma_m, t_p)$ must be a subtree of some tree in $\mathrm{comb}_\Sigma(L_1, L_2)$ and this means that $\varphi(\gamma_1) \cdots \varphi(\gamma_m) \in E(p, \sigma)$. (Note that $\varphi(\gamma_1) \cdots \varphi(\gamma_m) \notin E(p, \sigma)$ would mean that the computation of A on tree $\sigma(\gamma_1, \ldots, \gamma_m, t_p)$ would become blocked after processing the children of the root.)

We have shown that $\varphi(L_1) \subseteq E(p, \sigma)$. The other inclusion follows directly from the definition of the comb tree languages because any sequence of leaves that are children of a node of the right branch must spell out a string of L_1. □

Lemma 2 says, roughly speaking, that for any DTA(DFA) A and any state p of A occurring on the right branch of some trees, the sequences of states that can occur at leaves labeling the siblings of a node where p occurs can be identified with the language L_1. For $\sigma \in \Sigma_2$ and $p \in Q_2$ we define the set of *σ-successors* of p as

$$\mathrm{Succ}(p, \sigma) = \{q \in Q_2 \mid (\exists w \in Q_1^*)\ wp \in \delta(q, \sigma)\}.$$

Now we can establish a lower bound for the sizes of horizontal DFAs of A based on the size of the smallest UFA for the language L_1. Recall that $H^A_{q,\sigma}$ is the DFA recognizing the horizontal language $\delta(q, \sigma)$.

Lemma 3. *Assume that L_1 is Σ_1-separating. Let $\sigma \in \Sigma_2$ and $p \in Q_2$ and assume that $E(p, \sigma) \neq \emptyset$. Then the language L_1 has an UFA of size*

$$\sum_{q \in \mathrm{Succ}(p,\sigma)} |H^A_{q,\sigma}|.$$

Proof. We note that $E(p, \sigma) \neq \emptyset$ implies $\mathrm{Succ}(p, \sigma) \neq \emptyset$. For $q \in \mathrm{Succ}(p, \sigma)$ we denote the DFA $H^A_{q,\sigma}$ as $(P_q, Q, \gamma_q, i_q, F_q)$, where without loss of generality $P_{q_1} \cap P_{q_2} = \emptyset$ when $q_1 \neq q_2$. From the definition of $E(p, \sigma)$ we get

$$E(p, \sigma) \cdot p = \left[\bigcup_{q \in \mathrm{Succ}(p,\sigma)} \delta(q, \sigma) \right] \cap Q_1^* \cdot p.$$

We define an NFA $B = (S_B, Q, \delta_B, I_B, F_B)$ where $S_B = \bigcup_{q \in \mathrm{Succ}(p,\sigma)} P_q$, $I_B = \{i_q \mid q \in \mathrm{Succ}(p, \sigma)\}$, and

$$F_B = \{s \in P_q \mid q \in \mathrm{Succ}(p, \sigma),\ \ \gamma_q(s, p) \in F_q\}.$$

The final states of B are all states of the DFAs $H^A_{q,\sigma}$ that the symbol p takes to a final state. Finally the transitions of B are defined by setting, for $r \in Q$ and $s \in P_q$, where $q \in \mathrm{Succ}(p, \sigma)$,

$$\delta_B(s, r) = \gamma_q(s, r).$$

The transition relation of B is single valued and B is nondeterministic only because it has multiple initial states. Clearly B recognizes the language $E(p, \sigma)$. If B were ambiguous, then for distinct states $q_1, q_2 \in \mathrm{Succ}(p, \sigma)$ and for some $w \in Q_1^*$ both DFAs $H^A_{q_1,\sigma}$ and $H^A_{q_2,\sigma}$ would need to accept wp. This is impossible because A is deterministic. The claim follows since, by Lemma 2, when $E(p, \sigma) \neq \emptyset$, it is isomorphic to L_1. □

The following lemma establishes that, as should be expected, the number of states of Q_2 cannot be smaller than the size of the minimal DFA for the string language L_2.

Lemma 4. *The language L_2 is recognized by a DFA of size at most $|Q_2|$.*

Proof. We define a DFA $C = (Q_2, \Sigma_2, \delta_C, q_{0,C}, F_C)$ where $q_{0,C}$ is the unique element $q \in Q_2$ such that $\varepsilon \in \delta(q, \#_\Sigma)$ and $F_C = Q_2 \cap F$. The transition relation δ_C is defined by setting $\delta_C(q, \sigma)$ to be an arbitrary, but fixed, element of $\mathrm{Succ}(q, \sigma)$, for $\sigma \in \Sigma_2$ and $q \in Q_2$.

The computation of C on $w \in \Sigma_2^*$ simulates a computation of A on a right branch w of some tree t. The initial state of C is the state assigned by A to the special symbol $\#_\Sigma$ labeling the rightmost leaf of any accepted tree. In general, the computation of A depends in addition to the right branch also on the leaf labels that spell out the child-string of the current node. However, using induction on the length of w it is easy to see that C reaches the end of w in state q if and only if there exists a tree t with $\mathrm{rstring}(t) = w$ and A reaches the root of t in state q.

Since the right-strings of trees accepted by A spell out exactly the strings of L_2, the choice of final states of C implies the claim. □

Note that if A is v-minimal, it is easy to verify that for each $q \in Q_2$ and $\sigma \in \Sigma_2$, $\mathrm{Succ}(q, \sigma)$ has cardinality at most one and in this case the minimal DFA for L_2 has exactly Q_2 states. However, a v-minimal DTA(DFA) does often not have the smallest total number of states [13,19] and our lower bound result needs to allow that A has redundant (pairwise equivalent) vertical states.

For our lower bound result we still need one more technical lemma. Let $C = (Q_2, \Sigma_2, \delta_C, q_{0,C}, F_C)$ be the DFA for the language L_2 constructed in the proof of Lemma 4. Denote by $\equiv_C \subseteq Q_2 \times Q_2$ the equivalence relation that merges equivalent states of C and let

$$D = (Q_2/_{\equiv_C}, \Sigma_2, \delta_D, q_{0,D}, F_D) \tag{6}$$

be the minimized DFA where the states consist of $\equiv_C$-classes [26,28].

Lemma 5. *For any $\sigma \in \Sigma_2$ and $p_1, p_2 \in Q_2$ such that*

$$[p_1]_{\equiv_C} \not\sim_{D,\sigma} [p_2]_{\equiv_C} \tag{7}$$

we have $\mathrm{Succ}(p_1, \sigma) \cap \mathrm{Succ}(p_2, \sigma) = \emptyset$.

Proof. Choose $w_i \in \Sigma_2^*$ such that the computation of C on w_i reaches state p_i, $i = 1, 2$. For the sake of contradiction assume that $q \in \mathrm{Succ}(p_1, \sigma) \cap \mathrm{Succ}(p_2, \sigma)$. Thus, there exist $t_1, t_2 \in T_\Sigma$ such that $\mathrm{rstring}(t_i) = w_i\sigma$, $i = 1, 2$, and

$$t_1^A = t_2^A = q. \tag{8}$$

Since D is the minimal DFA for L_2, the assumption (7) implies that there exists $z \in \Sigma_2^*$ such that $w_1\sigma z \in L_2$ and $w_2\sigma z \notin L_2$ (or vice versa). Choose a tree $t_0 \in T_\Sigma$ where $\mathrm{rstring}(t_0) = z$, and for each $u \in \mathrm{RB}(t_0)$, $\mathrm{cstring}(t_0, u) \in L_1$. Denote the rightmost node of t_0 by u_{right} and let $t_i' = t_0(u_{\mathrm{right}} \leftarrow t_i)$, $i = 1, 2$. We note that $\mathrm{rstring}(t_i') = w_i\sigma z$ and for each $u \in \mathrm{RB}(t_i')$, $\mathrm{cstring}(t_i', u) \in L_1$. Thus, $t_1' \in \mathrm{comb}_\Sigma(L_1, L_2)$ and $t_2' \notin \mathrm{comb}_\Sigma(L_1, L_2)$ which contradicts (8). □

The following corollary puts together the results of Lemmas 2–5.

Corollary 6. *Let $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \{\#_\Sigma\}$ and $L_i \subseteq \Sigma_i^*$ is a regular language, $i = 1, 2$, where L_1 is Σ_1-separating. Let $A = (Q, \Sigma, \delta, F)$ be an arbitrary DTA(DFA) recognizing the tree language* $\mathrm{comb}_\Sigma(L_1, L_2)$. *Then*

$$\mathrm{size}(A) \geq \left[\mathrm{sc}(L_2) + |\Sigma_1|;\ \left(\sum_{\sigma \in \Sigma_2} \#_\sigma(L_2)\right) \cdot \mathrm{usc}(L_1) \right].$$

Proof. In the following $Q_1, Q_2 \subseteq Q$ are as in (4). The lower bound on the number of vertical states follows from Lemma 4 together with the observation, made in the proof of Lemma 2, that A must assign a different state of Q_1 to each leaf labeled by a symbol of Σ_1.

Consider arbitrary $\sigma \in \Sigma_2$ and $p \in Q_2$. Let D as in (6) be the minimal DFA for L_2 whose states are equivalence classes of Q_2. Now if $\delta_D([p]_{\equiv_C}, \sigma)$ is defined, the state p (of A) must have σ-successors and hence $E(p, \sigma) \neq \emptyset$, which means, by Lemma 2, that $E(p, \sigma)$ is isomorphic to L_1.

Furthermore, by Lemma 5, there exists a set $P \subseteq Q$ of cardinality $\#_\sigma(L_2)$ such that for each $p \in P$, $\delta_D([p]_{\equiv_C}, \sigma)$ is defined and for any $p_1, p_2 \in P$, $p_1 \neq p_2$, the sets $\mathrm{Succ}(p_1, \sigma)$ and $\mathrm{Succ}(p_2, \sigma)$ are disjoint. This means that the collections of horizontal DFAs of A that, in Lemma 3, are used to construct the UFA for each of the languages $E(p, \sigma)$, $p \in P$, are disjoint. Thus the sum of the sizes of horizontal DFAs of A associated with the symbol σ is at least $\#_\sigma(L_2) \cdot \mathrm{usc}(L_1)$. □

4. State complexity of determinization of unranked tree automata

The following upper bound for the size blow-up of converting an NTA(NFA) to a DTA(DFA) was given in [18].

Proposition 7 ([18]). *Suppose $A = (Q, \Sigma, \delta, F)$ is an NTA(NFA) and for $q \in Q$, $\sigma \in \Sigma$, denote the number of states of the horizontal NFA $H_{q,\sigma}^A$ by $m_{q,\sigma}$. Then there exists a DTA(DFA) B equivalent to A where*

$$\mathrm{size}(B) \leq \left[2^{|Q|} - 1;\ (2^{|Q|} - 1) \cdot \left(\sum_{\sigma \in \Sigma} 2^{\left(\sum_{q \in Q} m_{q,\sigma}\right)} - 1 \right) \right]. \tag{9}$$

The bound (9) follows from the construction of Lemma 4.4 of [18] by taking into account that the vertical computation of B does not need the dead state and, similarly, for each of the horizontal DFAs of B we can omit the dead state. (The statement of Lemma 4.4 in [18] uses a simplified formula and the possibility to omit the dead states is mentioned in the conclusion of [18].)

It remains open whether the upper bound of Proposition 7 can be reached. The paper [18] gives for the number of horizontal states only a lower bound based on an ad hoc construction that uses particular types of unary languages for which it is possible to establish that there cannot be trade-offs between the numbers of vertical and horizontal states. Here using Corollary 6 and the known lower bounds for the NFA-to-UFA transformation due to Leung [12] we give a significantly improved lower bound for the determinization of an NTA(NFA).

Let $L_m^{\mathrm{Moore}} \subseteq \{a, b\}^*$ be the language recognized by the NFA with m states given in Fig. 1 and $L_n^{\mathrm{Leung}} \subseteq \{c, d\}^*$ is the language recognized by the NFA with n states given in Fig. 2, $m, n \geq 2$.

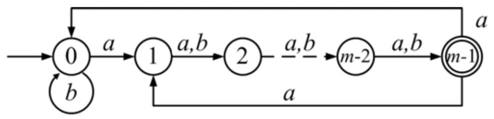

Fig. 1. An NFA A_{Moore} for the language L_m^{Moore}.

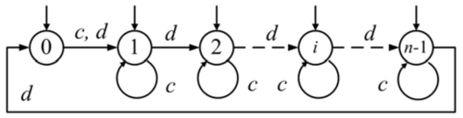

Fig. 2. An NFA A_{Leung} for the language L_n^{Leung}. The initial states are marked by incoming arrows and a state i is final iff i is odd.

Lemma 8. *Let $\Sigma = \{a, b, c, d, \#_\Sigma\}$ and $m, n \geq 2$. The tree language $\text{comb}_\Sigma(L_n^{\text{Leung}}, L_m^{\text{Moore}})$ has an NTA(NFA) A with*

$$\text{size}(A) = [m + 2; \quad (2m - 1)(n + 1) + 3].$$

Proof. Choose $A = (Q, \Sigma, \delta_A, \{m - 1\})$ where $Q = \{0, 1, \dots, m - 1, q_c, q_d\}$. For defining the transitions of A let $\varphi : \{c, d\}^* \to \{q_c, q_d\}^*$ be the morphism defined by setting $\varphi(c) = q_c, \varphi(d) = q_d$. Now the transitions of A are defined as

(i) $\delta_A(q_c, c) = \delta_A(q_d, d) = \delta_A(0, \#_\Sigma) = \varepsilon$,
(ii) $\delta_A(0, a) = \varphi(L_n^{\text{Leung}}) \cdot (m - 1)$,
(iii) $\delta_A(1, a) = \varphi(L_n^{\text{Leung}}) \cdot \{0, m - 1\}$,
(iv) $\delta_A(i, a) = \varphi(L_n^{\text{Leung}}) \cdot (i - 1), 2 \leq i \leq m - 1$.
(v) $\delta_A(0, b) = \varphi(L_n^{\text{Leung}}) \cdot 0$,
(vi) $\delta_A(i, b) = \varphi(L_n^{\text{Leung}}) \cdot (i - 1), 2 \leq i \leq m - 1$.

The NTA(NFA) uses $2m-1$ (nonempty) horizontal languages to define the transitions at the non-leaf nodes and each of these languages has an NFA with $n + 1$ states. Additionally, A needs three horizontal NFAs of size one to define the transitions at the leaf symbols. □

Note that, roughly, the same bound as below in Theorem 9, could be obtained for the tree languages $\text{comb}_\Sigma(L_n^{\text{Leung}}, L_m^{\prime\text{Leung}})$, where $L_m^{\prime\text{Leung}}$ is a copy of L_m^{Leung} defined over the alphabet $\{a, b\}$. We use here the languages L_m^{Moore} because for the second parameter of the comb tree languages it is sufficient that the NFA-to-DFA size blow-up is exponential (as opposed to the NFA-to-UFA size blow-up), and also because the construction, in any case, requires that the two string languages are defined over disjoint alphabets. Furthermore, substituting a copy of L_m^{Leung} for the language L_m^{Moore} would increase the number of horizontal states of the NTA(NFA) A in Lemma 8 by $m-1$ (because each initial state needs a horizontal NFA of size one associated with $\#_\Sigma$).

Combining Lemma 8 and Corollary 6 we can now state the main result.

Theorem 9. *There exists an NTA(NFA) with $m + 2$ vertical and $2m(n + 1) - n + 2$ horizontal states such that for any equivalent DTA(DFA) B*

$$\text{size}(B) \geq [2^m + 1; \quad (2^m + 2^{m-2} - 2) \cdot (2^n - 1)].$$

Proof. We choose A to be the NTA(NFA) of Lemma 8 recognizing $\text{comb}_\Sigma(L_n^{\text{Leung}}, L_m^{\text{Moore}})$, and hence $\Sigma_1 = \{c, d\}$, $\Sigma_2 = \{a, b\}$. From Theorem 3 of [12] we know that any UFA for L_n^{Leung} has at least $2^n - 1$ states.

Let C be the minimal incomplete DFA recognizing the language L_m^{Moore}. The DFA C is obtained from the NFA A_{Moore} by subset construction and the states of C are all nonempty subsets of $\{0, 1, \dots, m - 1\}$ [15]. It is easy to verify that

$$\#_a(C) = \frac{3}{4}2^m - 1, \qquad \#_b(C) = 2^{m-1} - 1.$$

Now by Corollary 6 we know that the total number of horizontal states of a DTA(DFA) equivalent to A is at least $(\#_a(C) + \#_b(C)) \cdot \text{usc}(L_n^{\text{Leung}})$ which gives the claimed lower bound. □

We make the following observations concerning the above lower bound construction. In order to avoid unnecessarily complicated notations, when defining the comb tree languages $\text{comb}_\Sigma(L_1, L_2)$, for each node of the right-branch, we have defined the set of child strings in terms of the same language L_1. By using a large number of languages to define the sets of child strings one could get a more "natural" lower bound as explained below.

Consider a DTA(DFA) $A = (Q, \Sigma, \delta, F)$ recognizing the language $\text{comb}_\Sigma(L_n^{\text{Leung}}, L_m^{\text{Moore}})$. The lower bound for the size of horizontal DFAs of A was obtained (in Corollary 6) by proving that for each $\sigma \in \Sigma_2$ and $p \in Q_2$ (with Q_2 as in (4)) the sum of the sizes of horizontal DFAs $H_{q,\sigma}^A$, $q \in \text{Succ}(p, \sigma)$ has to be at least as large as a minimal UFA for the language $E(p, \sigma)$. The disjoint union of languages recognized by DFAs $H_{q,\sigma}^A$, $q \in \text{Succ}(p, \sigma)$, recognizes, roughly speaking, an isomorphic copy of L_n^{Leung} catenated with individual states of Q_2. Thus, the collections of horizontal DFAs $H_{q,\sigma}^A$, $q \in \text{Succ}(p, \sigma)$ and $H_{q',\sigma}^A$,

$q' \in \mathrm{Succ}(p', \sigma)$ are similar and, at least on first sight, the lower bound could be viewed to be an artificial consequence of the requirement that A cannot reuse similar DFAs for horizontal languages corresponding to different vertical states.

However, using exactly the same idea as when defining the tree languages $\mathrm{comb}_{\Sigma}(L_1, L_2)$, we could define more complicated tree languages consisting of trees where the right-string is in L_2 and then the child-strings of each node u of the right branch would define a language L_{i_u}, where the index i_u would depend on the equivalence class (with respect to L_2) of the prefix of the right-string ending at node u. As long as each of the languages L_{i_u} needs a large UFA we would get a lower bound similar to the one given in Theorem 9. For example, Corollary 4 of [12] establishes that by, roughly speaking, reversing the design of the NFA A_{Leung} in Fig. 2 one obtains another family of NFAs that exhibits the worst-case NFA-to-UFA size blow-up. Other variants could be obtained, e.g., by changing the names of symbols, and one obtains an unlimited number of variants if we relax the requirement that the NFA-to-UFA size blow-up needs to be exactly $2^n - 1$.

As a corollary of Theorem 9 we have:

Corollary 10. *There exists an NTA(NFA) $A = (Q, \Sigma, \delta, F)$, over a fixed alphabet Σ, with $|Q| \in O(m)$ and where for each $q \in Q$, $\sigma \in \Sigma$ the horizontal language $\delta(q, \sigma)$ has an NFA of size $O(n)$ such that any equivalent DTA(DFA) needs $2^{\Omega(m)}$ vertical states and $2^{\Omega(m+n)}$ horizontal states.*

The lower bound for the number of horizontal states is considerably better than the lower bound from [18], however, it does not match the upper bound. In cases where the horizontal languages associated with different vertical states of the NTA(NFA) have roughly the same state complexity, Proposition 7 gives only an upper bound of $2^{O(m\cdot n)}$ for the horizontal size of the deterministic automaton.

5. Conclusion

We have given a lower bound for the size blow-up of determinizing a nondeterministic unranked tree automaton. The proof of the lower bound is based on the property that horizontal languages associated with a given alphabet symbol σ and different states need to be disjoint. This allows us to limit the state trade-off that could occur when a vertical state q is replaced by multiple equivalent states $q_1, \ldots, q_k$ using the observation that from DFAs associated with states $q_1, \ldots, q_k$ we can construct a UFA for the horizontal language of q.

A limitation of the method is that the UFA construction (as done in Lemma 3) works only in situations where (most of) the child nodes are leaves which guarantees that any DTA(DFA) has to assign a unique state to each such node.

If B is a DTA(DFA) constructed to simulate an NTA(NFA) $A = (Q, \Sigma, \delta, F)$ as in the proof of Proposition 7, the horizontal languages of B consist of strings over subsets of Q and, in order to guarantee that the vertical computation is deterministic, a horizontal DFA associated with $P \subseteq Q$ and $\sigma \in \Sigma$ needs[1] to simulate each horizontal NFA of A associated with σ. See [18] for a more detailed discussion on the construction of the proof of Proposition 7.

The above mentioned limitation in the technical construction of the proof of Lemma 3 prevents our lower bound from matching the upper bound of Proposition 7. It remains an open problem what is the precise worst-case size blow-up for the number of states of horizontal automata when converting a nondeterministic unranked tree automaton to a corresponding deterministic automaton.

References

[1] J.-C. Birget, Intersection and union of regular languages and state complexity, Inform. Process. Lett. 43 (1992) 185–190.
[2] A. Brüggemann-Klein, M. Murata, D. Wood, Regular tree and regular hedge languages over unranked alphabets, HKUST Technical report, 2001.
[3] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, C. Löding, S. Tison, M. Tommasi, Tree automata techniques and applications. Electronic book available at http://tata.gforge.inria.fr, 2007.
[4] J. Cristau, C. Löding, W. Thomas, Deterministic automata on unranked trees, in: Proc. of FCT'05, in: Lect. Notes Comput. Sci., vol. 3623, Springer, 2005, pp. 68–79.
[5] Z. Ésik, Y. Gao, G. Liu, S. Yu, Estimation of state complexity of combined operations, Theoret. Comput. Sci. 410 (2009) 3272–3280.
[6] F. Gécseg, M. Steinby, Tree languages, in [22], Vol. III, 1997, pp. 1–68.
[7] I. Glaister, J. Shallit, A lower bound technique of the size of nondeterministic finite automata, Inform. Proc. Lett. 59 (1996) 75–77.
[8] J. Goldstine, M. Kappes, C.M.R. Kintala, H. Leung, A. Malcher, D. Wotschke, Descriptional complexity of machines with limited resources, J. Universal Comput. Sci. 8 (2002) 193–234.
[9] M. Holzer, M. Kutrib, Nondeterministic finite automata — recent results on the descriptional and computational complexity, Internat. J. Foundations Comput. Sci. 20 (2009) 563–580.
[10] M. Holzer, M. Kutrib, Descriptional and computational complexity of finite automata — a survey, Inf. Comput. 209 (2011) 456–470.
[11] J. Hromkovič, Communication Complexity and Parallel Computing, Springer-Verlag, 1997.
[12] H. Leung, Descriptional complexity of NFA of different ambiguity, Internat. J. Foundations Comput. Sci. 16 (5) (2005) 975–984.
[13] W. Martens, J. Niehren, On the minimization of XML schemas and tree automata for unranked trees, J. Comput. System Sci. 73 (2007) 550–583.
[14] T. Milo, D. Suciu, V. Vianu, Typechecking for XML transformers, J. Comput. System Sci. 66 (2003) 66–97.
[15] F.R. Moore, On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic, and two-way finite automata, IEEE Transactions on Computers 20 (1971) 1211–1214.
[16] F. Neven, Automata theory for XML researchers, SIGMOD Record 31 (2002) 39–46.
[17] A. Okhotin, Unambiguous finite automata over a unary alphabet, in: Mathematical Foundations of Computer Science, MFCS 2010, in: Lect. Notes Comput. Sci., vol. 6281, Springer, 2010, pp. 556–567.
[18] X. Piao, K. Salomaa, Transformations between different models of unranked bottom-up tree automata, Fund. Informaticae 109 (2011) 405–424.

[1] That is, at least there is no straightforward way to avoid this.

[19] X. Piao, K. Salomaa, State trade-offs in unranked tree automata, in: Proc. 13th Workshop Descriptional Complexity of Formal Systems, DCFS'11, in: Lect. Notes Comput. Sci., vol. 6808, Springer, 2011, pp. 261–274.
[20] X. Piao, K. Salomaa, State complexity of Kleene-star operations on trees, in: Proc. of WTCS 2012, in: Lect. Notes Comput. Sci., vol. 7160, Springer, 2012, pp. 388–402.
[21] S. Raeymaekers, M. Bruynooghe, Minimization of finite unranked tree automata, manuscript, 2004.
[22] G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, Vol. I–III, Springer-Verlag, 1997.
[23] A. Salomaa, K. Salomaa, S. Yu, Undecidability of the state complexity of composed regular operations, in: Proc. LATA 2011, in: Lect. Notes Comput Sci., vol. 6638, Springer, 2011, pp. 489–498.
[24] E.M. Schmidt, Succinctness of description of context-free, regular and unambiguous languages, Ph.D. thesis. Cornell University, 1978.
[25] T. Schwentick, Automata for XML, J. Comput. System Sci. 73 (2007) 289–315.
[26] J. Shallit, A Second Course in Formal Languages and Automata Theory, Cambridge University Press, 2009.
[27] R.E. Stearns, H.B. Hunt III, On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata, SIAM J. Comput. 14 (1985) 598–611.
[28] S. Yu, Regular languages, in [22], Vol. I, 1997, pp. 41–110.