Technical Report No. 2010-567
State Complexity of Unranked Tree Automata

Xiaoxue Piao and Kai Salomaa

School of Computing, Queen's University
Kingston, Ontario K7L 3N6, Canada
{piao,ksalomaa}@cs.queensu.ca

Abstract. We consider the representational state complexity of unranked tree automata. The bottom-up computation of an unranked tree automaton may be either deterministic or nondeterministic, and further variants arise depending on whether the horizontal string languages defining the transitions are represented by a DFA or an NFA. Also, we consider for unranked tree automata the alternative syntactic definition of determinism introduced by Cristau et al. We establish upper and lower bounds for the state complexity of conversions between different types of unranked tree automata.

Key words: tree automata, unranked trees, state complexity, nondeterminism

1 Introduction

Descriptional complexity, or state complexity, of finite automata has been extensively studied in recent years, see [6, 8, 12, 14] and references listed there. On the other hand, very few papers explicitly discuss state complexity of tree automata. For classical tree automaton models operating on ranked trees [4, 5] many state complexity results are similar to corresponding results on string automata. For example, it is well known that determinizing an n state nondeterministic bottom-up tree automaton gives an automaton with at most 2^n states. Similarly, for Boolean operations or the extension of concatenation and Kleene star to trees the state complexity results would be similar to the corresponding results for ordinary finite automata [14].

Modern applications of tree automata, such as XML document processing [10, 13], use automata operating on unranked trees. In some work the unranked trees are first encoded as binary trees [2], however, a more common and conceptually cleaner approach is to define the computation of the tree automaton directly on unranked XML-trees [1, 4, 13]. The set of transitions of an unranked tree automaton is, in general, infinite and the transitions are usually specified in terms of a regular language. Thus, in addition to the finite set of states used in the bottom-up computation, an unranked tree automaton needs for each state q and input symbol σ a finite string automaton to recognize the *horizontal language* consisting of strings of states defining the transitions associated to q and σ.

Here we consider bottom-up (frontier-to-root) unranked tree automata. Roughly speaking, we get different models depending on whether the bottom-up computation is nondeterministic or deterministic and whether the horizontal languages are recognized by an NFA or

a DFA ((non-)deterministic finite automaton). Furthermore, there are two essentially different ways to define determinism for unranked tree automata. The more common definition [4, 13] requires that for any input symbol σ and two distinct states q_1, q_2, the horizontal languages associated, respectively, with q_1 and σ and with q_2 and σ are disjoint. The condition guarantees that the bottom-up computation assigns a unique state to each node. To distinguish this from the syntactic definition of determinism of [3], we call a deterministic tree automaton where the horizontal languages defining the transitions are specified by DFAs, a *weakly deterministic tree automaton.* Note that a computation of a weakly deterministic automaton still needs to "choose" which of the DFAs (associated with different states) is used to process the sequence of states that the computation reached at the children of the current node – since the intersection of distinct horizontal languages is empty the choice is unambiguous, however, when beginning to process the sequence of states the automaton has no way of knowing which DFA to use.

A different definition, that we call *strong determinism,* was introduced by Cristau, Löding and Thomas [3].[1] A strongly deterministic automaton associates to each input symbol a single DFA H_σ equipped with an output function, and the state H_σ reaches after processing the sequence of states corresponding to the children determines (via the output function) the state at a parent node labeled by σ. Strongly deterministic automata can be minimized efficiently and the minimal automaton is unique [3]. On the other hand, interestingly it was shown by Martens and Niehren [9] that for weakly deterministic tree automata the minimization problem is NP-complete and the minimal automaton need not be unique.

We study the state complexity of determinizing different variants of nondeterministic tree automata, that is, we develop upper and lower bounds for the size of deterministic tree automata that are equivalent to given nondeterministic automata. We define the size of an unranked tree automaton as a pair of integers consisting of the number of states used in the bottom-up computation, and the sum of the sizes of the NFAs definining the horizontal languages. Note that the two types of states play very different roles in computations of the tree automaton. The other possibility would be, as is done e.g. in [9], to count simply the total number of all states in the different components.

Also, we study the state complexity of the conversions between the strongly and the weakly deterministic tree automata. Although the former model can be viewed to be more restricted, there exist tree languages for which the size of a strongly deterministic automaton is smaller than the size of the minimal weakly deterministic automaton. It turns out to be more difficult to establish lower bounds for the size of weakly deterministic automata than is the case for strongly deterministic automata. Naturally, this can be expected due to the intractability of the minimization of weakly deterministic automata [9].

To conclude we summarize the contents of the paper. In section 2 we recall definitions for tree automata operating on unranked trees and introduce some notation. In section 3

[1] The paper [3] refers to weak and strong determinism, respectively, as semantic and syntactic determinism.

we study the descriptional complexity of conversions between the strongly and the weakly deterministic tree automata, and in section 4 we study the size blow-up of converting different variants of nondeterministic tree automata to strongly and weakly deterministic automata, respectively.

2 Preliminaries

We assume that the reader is familiar with the basics of formal languages and finite automata [7, 14]. Below we briefly recall some definitions for tree automata operating on unranked trees and fix notations. More details on unranked tree automata and references can be found in [4, 13]. A general reference on tree automata operating on ranked trees is [5].

Basic notions concerning trees, such as the root, a leaf, a subtree, the height of a tree and children of a node are assumed to be known. The set of non-negative integers is $\mathbb{N}$. A *tree domain* is a prefix-closed subset D of $\mathbb{N}^*$ such that if $ui \in D$, $u \in \mathbb{N}^*$, $i \in \mathbb{N}$ then $uj \in D$ for all $j < i$. The set of nodes of a tree t is represented in the well-known way as a tree domain $\mathrm{dom}(t)$ and t is a mapping $\mathrm{dom}(t) \to \Sigma$ where Σ is a finite alphabet of symbols. Thus, we use labeled ordered unranked trees. Each node of a tree has a finite number children with a linear order, but there is no a priori upper bound on the number of children of a node. The set of all Σ-labeled trees is T_Σ.

We introduce following notation for trees. For $i \geq 0$, $a \in \Sigma$ and $t \in T_\Sigma$, we denote by $a^i(t) = a(a(...a(t)...))$ a tree, where the nodes ε, 1, ..., 1^{i-1} are labelled by a and the subtree at node 1^i is t. When $a \in \Sigma$, $w = b_1b_2...b_n \in \Sigma^*$, $b_i \in \Sigma$, $1 \leq i \leq n$, we use $a(w)$ to denote the tree $a(b_1, b_2, ..., b_n)$. When L is a set of strings, $a(L) = \{a(w) \mid w \in L\}$. The set of all Σ-trees where exactly one leaf is labelled by a special symbol x ($x \notin \Sigma$) is $T_\Sigma[x]$. For $t \in T_\Sigma[x]$ and $t' \in T_\Sigma$, $t(x \leftarrow t')$ denotes the tree obtained from t by replacing the unique occurrence of variable x by t'.

A *nondeterministic (unranked) tree automaton* (NTA) is a tuple $A = (Q, \Sigma, \delta, F)$, where Q is the finite set of states, Σ is the alphabet labeling nodes of input trees, $F \subseteq Q$ is the set of final states, and δ is a mapping from $Q \times \Sigma$ to the subsets of Q^* which satisfies the condition that, for each $q \in Q$, $\sigma \in \Sigma$, $\delta(q, \sigma)$ is a regular language. The language $\delta(q, \sigma)$ is called the *horizontal language* associated with q and σ.

A computation of A on a tree $t \in T_\Sigma$ is a mapping $C : \mathrm{dom}(t) \to Q$ such that for $u \in \mathrm{dom}(t)$, if $u \cdot 1, \ldots u \cdot m$, $m \geq 0$, are the children of u then $C(u \cdot 1) \cdots C(u \cdot m) \in \delta(C(u), t(u))$. In case u is a leaf the condition means that $m = 0$ and $\varepsilon \in \delta(C(u), t(u))$.

Intuitively, if a computation of A has reached the children of a σ-labelled node u in a sequence of states $q_1, q_2, \ldots, q_m$, the computation may nondeterministically assign a state q to the node u provided that $q_1q_2 \cdots q_m \in \delta(q, \sigma)$. For $t \in T_\Sigma$, $t^A \subseteq Q$ denotes the set of states that in some bottom-up computation A may reach at the root of t. The *tree language recognized by* A is defined as $L(A) = \{t \in T_\Sigma \mid t^A \cap F \neq \emptyset\}$.

For a tree automaton $A = (Q, \Sigma, \delta, F)$, we denote by $H^A_{q,\sigma}$, $q \in Q$, $\sigma \in \Sigma$, a nondeterministic finite automaton (NFA) on strings recognizing the horizontal language $\delta(q, \sigma)$. The NFA $H^A_{q,\sigma}$ is called a horizontal automaton, and states of different horizontal automata are called collectively *horizontal states.* We refer to the states of Q that are used in the bottom-up computation as *vertical states.*

A tree automaton $A = (Q, \Sigma, \delta, F)$ is said to be (semantically) *deterministic* (a DTA) if for $\sigma \in \Sigma$ and any two states $q_1 \neq q_2$, $\delta(q_1, \sigma) \cap \delta(q_2, \sigma) = \emptyset$.

We get a further refinement of classes of automata depending on whether the horizontal languages are defined using DFAs or NFAs. We use NTA(M) or DTA(M), respectively, to denote (the class of) nondeterministic or deterministic tree automata where the horizontal languages are recognized by the elements in class M. For example, NTA(DFA) denotes the tree automata where the horizontal languages are recognized by a DFA.

Note that when referring to a tree automaton $A = (Q, \Sigma, \delta, F)$ it is always assumed that the relation δ is specified in terms of automata $H^A_{q,\sigma}$, $q \in Q$, $\sigma \in \Sigma$, and by saying that A is an NTA(DFA) we indicate that each $H^A_{q,\sigma}$ is a DFA. We refer to DTA(DFA)'s also as *weakly deterministic tree automata* to distinguish them from the below notion of strong determinism.

If A is a DTA(NFA), for any tree $t \in T_\Sigma$ the bottom-up computation of A assigns a unique vertical state to the root of t, that is, t^A is a singleton set or empty. If the horizontal automata $H^A_{q,\sigma}$ are DFAs, furthermore, for each transition the sequence of horizontal states is processed deterministically. However, as discussed in section 1, a computation that has reached children of a σ-labeled node in a sequence of states $w \in Q^*$ still needs to make the choice which of the DFAs $H^A_{q,\sigma}$, $q \in Q$, is used to process w. For this reason we consider also the following notion introduced in [3] that we call strong determinism.

A tree automaton $A = (Q, \Sigma, \delta, F)$ is said to be *strongly deterministic* if for each $\sigma \in \Sigma$, the transitions are defined by a single DFA augmented with an output function as follows. For $\sigma \in \Sigma$ define

$$H^A_\sigma = (S_\sigma, Q, s^0_\sigma, F_\sigma, \gamma_\sigma, \lambda_\sigma), \tag{1}$$

where $(S_\sigma, Q, s^0_\sigma, F_\sigma, \gamma_\sigma)$ is a DFA with set of states S_σ ($s^0_\sigma \in S_\sigma$ is the start state, $F_\sigma \subseteq S_\sigma$ is the set of final states and $\gamma_\sigma : S_\sigma \times Q \to S_\sigma$ is the transition function) and λ_σ is a function $F_\sigma \to Q$. Then we require that for all $q \in Q$ and $\sigma \in \Sigma$: $\delta(q, \sigma) = \{w \in Q^* \mid \lambda_\sigma(\gamma_\sigma(s^0_\sigma, w)) = q\}$.[2] Note that the definition guarantees that $\delta(q_1, \sigma) \cap \delta(q_2, \sigma) = \emptyset$ for any distinct $q_1, q_2 \in Q$, $\sigma \in \Sigma$. The class of strongly deterministic tree automata is denoted as SDTA.

By the size of an NFA B, denoted size(B), we mean the number of states of B. Because the roles played by vertical and horizontal states, respectively, in the computations of a tree automaton are essentially different, when measuring the size of an automaton we count the

[2] Strictly speaking, δ is superfluous in the tuple specifying an SDTA and the original definition of [3] gives instead the automata H^A_σ, $\sigma \in \Sigma$. We use δ in order to make the notation compatible with our other models, and to avoid having to define bottom-up computations of SDTAs separately.

two types of states separately. The size of an NTA(NFA) $A = (Q, \Sigma, \delta, F)$ is defined as

$$\text{size}(A) = [\ |Q|;\ \sum_{q\in Q, \sigma\in\Sigma} \text{size}(H^A_{q,\sigma})\ ]\ \ (\in \mathbb{N}\times\mathbb{N}).$$

Using notations of (1), the size of an SDTA A is defined as the pair of integers $\text{size}(A) = [\ |Q|;\ \sum_{\sigma\in\Sigma} |S_\sigma|\ ]$.

We make the following notational convention that allows us to use symbols of Σ in the definition of horizontal languages. Unless otherwise mentioned, we assume that a tree automaton always assigns to each leaf symbol labeled σ a state $\overline{\sigma}$ that is not used anywhere else in the computation. That is, for $\sigma \in \Sigma$ and $q \in Q$, $\varepsilon \in \delta(q, \sigma)$ only if $q = \overline{\sigma}$, $\delta(\overline{\sigma}, \sigma) = \{\varepsilon\}$ and $\delta(\overline{\tau}, \sigma) = \emptyset$ for all $\sigma, \tau \in \Sigma$, $\sigma \neq \tau$. When there is no confusion, we denote also $\overline{\sigma}$ simply by σ. When the alphabet Σ is fixed, there is only a constant number of the special states $\overline{\sigma}$ and since, furthermore, the special states have the same function in all types of tree automata, for simplicity, we do not include them when counting the vertical states. The purpose of this convention is to improve readability: many of our constructions become more transparent when alphabet symbols can be used explicitly to define horizontal languages. The convention does not change our state complexity bounds that are generally given within a multiplicative constant.

To conclude this section we give two lemmas that provide lower bound estimates for vertical and horizontal states of SDTAs, respectively. The lower bound condition for vertical states applies, more generally, for DTA(NFA)'s, however, obtaining lower bounds for the number of horizontal states of weakly deterministic automata turns out to be more problematic.

Lemma 1. *Let A be an SDTA or a DTA(NFA) with a set of vertical states Q recognizing a tree language L. Assume $R = \{t_1, \ldots, t_m\} \subseteq T_\Sigma$ where for any $1 \leq i < j \leq m$ there exists $t \in T_\Sigma[x]$ such that $t(x \leftarrow t_i) \in L$ iff $t(x \leftarrow t_j) \notin L$. Then $|Q| \geq |R| - 1$.*

Proof. The condition of the lemma guarantees that the state of A can be undefined at the root of at most one of the trees of R. If Q has less than $|R| - 1$ states, then A must reach in the same state the root of two distinct trees $t_1, t_2 \in R$. According to the lemma, there exists $t \in T[x]$ such that $t(x \leftarrow t_1)$ if and only if $t(x \leftarrow t_2)$. This is a contradiction. ■

Lemma 2. *Let A be an SDTA with a set of vertical states Q recognizing a tree language L. Let S be a finite set of tuples of Σ-trees and $b \in \Sigma$. Assume that for any distinct tuples $(r_1, \ldots, r_m)$, $(s_1, \ldots, s_n) \in S$ there exists $t \in T_\Sigma[x]$ and a sequence of trees $u_1, \ldots, u_k$ such that*

$$t(x \leftarrow b(r_1, \ldots, r_m, u_1, \ldots, u_k)) \in L \ \ \textit{iff}\ \ t(x \leftarrow b(s_1, \ldots, s_n, u_1, \ldots, u_k)) \notin L \qquad (2)$$

Then the horizontal automaton H^A_b needs at least $|S| - 1$ states.

Proof. If H_b^A has less than $|S|-1$ states, then for two distinct tuples $(r_1, \ldots, r_m)$, $(s_1, \ldots, s_n)$ of S the automaton H_b^A must be in the same state after reading the strings $r_1^A \cdots r_m^A$ and $s_1^A \cdots s_n^A$ ($\in Q^*$). Note that the condition (2) guarantees that at most one tuple of S can contain a tree r for which r^A is undefined. Now A reaches the same vertical state at roots of $t(x \leftarrow b(r_1, \ldots, r_m, u_1, \ldots, u_k))$ and $t(x \leftarrow b(s_1, \ldots, s_n, u_1, \ldots, u_k))$. This contradicts (2). ■

3 Size comparison of the strongly and weakly deterministic tree automata

Here we give upper and lower bounds for the size of a weakly deterministic automaton (a DTA(DFA)) simulating a strongly deterministic one (an SDTA), and vice versa. The computation of a DTA(DFA) can, in some sense, nondeterministically choose which of the horizontal DFAs it uses at each transition. An SDTA does not have this capability and it can be expected that, in the worst case, an SDTA may need considerably more states than an equivalent DTA(DFA). However, there exist also tree languages for which an SDTA can be considerably more succinct than a DTA(DFA).

3.1 Converting an SDTA to a DTA(DFA)

First we give an upper bound for the conversion. In the below lemma (and afterwards) we use "$\leq$" to compare pairs of integers componentwise. As introduced in section 2, for an SDTA A we denote the deterministic automata for the corresponsing horizontal languages by H_σ^A, $\sigma \in \Sigma$.

Lemma 3. *Let $A = (Q, \Sigma, \delta, F)$ be an arbitrary SDTA.*

We can construct an equivalent DTA(DFA) A' where

$$\text{size}(A') \leq [\ |Q|;\ |Q| \times \sum_{\sigma \in \Sigma} \text{size}(H_\sigma^A)\ ]. \tag{3}$$

Proof. For $\sigma \in \Sigma$ denote components of H_σ^A as in (1). Construct an equivalent DTA(DFA) $A' = (Q, \Sigma, \delta', F)$, where for each $\sigma \in \Sigma$, $q \in Q$, $\delta'(q, \sigma) = \{w \in Q^* \mid \lambda_\sigma(\gamma_\sigma(s_\sigma^0, w)) = q\}$. The languages $\delta'(q_1, \sigma)$ and $\delta'(q_2, \sigma)$, $q_1 \neq q_2$ are always disjoint, and $\delta'(q, \sigma)$ is recognized by a DFA obtained from H_σ^A by choosing as the set of final states $\lambda_\sigma^{-1}(q)$, $q \in Q$, $\sigma \in \Sigma$. The construction does not change the number of vertical states and (3) holds. ■

Next we establish a lower bound for the conversion.

Lemma 4. *Let $n, z \in \mathbb{N}$ and choose $\Sigma = \{a, b, 0, 1\}$. There exists an SDTA B with input alphabet Σ, n vertical states and $z + 4n$ horizontal states, such that any DTA(DFA) for the tree language $L(B)$ has at least n vertical states and $n(\lfloor \log n \rfloor + 2 + z)$ horizontal states.*

Proof. Let $n, z \geq 1$ be arbitrary but fixed. For $1 \leq i \leq n$, $y_i \in \{0,1\}^*$ is the binary representation of i. Define $L = \{a^i(b^z y_i) \mid n \geq i \geq 1\}$.

The tree language L is recognized by an SDTA $B = (Q, \Sigma, \delta, F)$, where $Q = \{q_1, ..., q_n\}$, $F = \{q_1\}$, $\delta(a, q_i) = b^z \cdot y_i + q_{i+1}$, when $1 \leq i \leq n-1$, and, $\delta(a, q_n) = b^z \cdot y_n$.[3]

Clearly the bottom-up computations of B recognize the tree language L and it remains to estimate the size of the DFA with output H_a^B that defines transitions at a-labeled nodes. The DFA needs $z+1$ states to process the prefix b^z and at most $2^{(2+\lfloor \log n \rfloor)} - 1$ states to remember the suffix of length $1 + \lfloor \log n \rfloor$ and output the correct vertical state using the λ-function. Thus, B can be constructed with $z + 4n$ horizontal states.

Consider an arbitrary DTA(DFA) $B' = (Q', \Sigma, \delta', F')$ accepting L. First using Lemma 1 we see that $|Q'| \geq n$. Choose $R = \{a(b^z y_i) \mid 1 \leq i \leq n\} \cup \{a(b^{z+1})\}$. Clearly for any $t_1, t_2 \in R$ there exists $t \in T_\Sigma[x]$ such that $t(x \leftarrow t_1) \in L$ if and only if $t(x \leftarrow t_2) \notin L$.

Denote by p_i the state that B' assigns to the root of $a(b^z y_i)$, $1 \leq i \leq n$. It is easy to verify (as in Lemma 1) that $p_i \neq p_j$ when $i \neq j$. Now $\delta'(p_i, a) \cap \Sigma^* = \{b^z y_i\}$. (If $\delta'(p_i, a)$ were to contain some other string over Σ, B' will accept trees not in L.) Thus the DFA $H_{p_i,a}^{B'}$ recognizing $\delta'(p_i, a)$ has at least $|b^z y_i| + 1 = z + \lfloor \log n \rfloor + 2$ states.

Since the above holds for all $1 \leq i \leq n$, a lower bound for the numbers of vertical and horizontal states of B' is size$(B') \geq [\, n;\ n \cdot (\lfloor \log n \rfloor + 2 + z)\,]$ ■

Using Lemma 4 with $z = n - \lfloor \log n \rfloor$, we see that the upper bound of Lemma 3 is tight within a multiplicative contant. This is stated as:

Theorem 1. *An SDTA with n vertical and m horizontal states can be simulated by a DTA(DFA) having n vertical and $n \cdot m$ horizontal states.*

For $n \geq 1$, there exists a tree language L_n recognized by an SDTA with n vertical and $O(n)$ horizontal states such that any DTA(DFA) recognizing L_n has n vertical and $\Omega(n^2)$ horizontal states.

It can be viewed as expected that in the conversion of Theorem 1 the number of vertical states does not change. However as will be discussed later, in general, for a DTA(DFA) it may be possible to reduce the number of horizontal states by increasing the number of vertical states.

3.2 Converting a DTA(DFA) to an SDTA

Again we give first an upper bound for the simulation.

Lemma 5. *Let $B = (Q, \Sigma, \delta, F)$ be an arbitrary DTA(DFA), where $|Q| = n$. Let $H_{q,\sigma}^B = (S_{q,\sigma}, Q, s_{q,\sigma}^0, F_{q,\sigma}, \gamma_{q,\sigma})$ be a DFA for the horizontal language $\delta(q, \sigma)$, $q \in Q$, $\sigma \in \Sigma$.*

[3] As explained in section 2, we use notation where an alphabet symbol σ occuring in strings of a horizontal language is interpreted as a leaf node labeled by σ.

We can construct an equivalent SDTA B' where

$$\text{size}(B') \leq [\ |Q|;\ \sum_{\sigma\in\Sigma}(\prod_{q\in Q}(|S_{q,\sigma}| - |F_{q,\sigma}|) + \sum_{q\in Q}|F_{q,\sigma}| \cdot \prod_{p\in Q, p\neq q}(|S_{p,\sigma}| - |F_{p,\sigma}|))\ ]$$

.

Proof. We construct an equivalent SDTA $B' = (Q, \Sigma, \delta', F)$ as follows. Denote $Q = \{q_1, \ldots, q_n\}$. For each $\sigma \in \Sigma$, we define a DFA with output

$$H_\sigma^{B'} = (\prod_{q\in Q} S_{q,\sigma}, Q, (s^0_{q_1,\sigma}, \ldots, s^0_{q_n,\sigma}), E_\sigma, \Delta_\sigma, \lambda_\sigma),$$

where for $p_i \in S_{q_i,\sigma}$, $1 \leq i \leq n$,

$$\Delta_\sigma((p_1, p_2, \ldots, p_n), q) = (\gamma_{q_1,\sigma}(p_1, q), \ldots, \gamma_{q_n,\sigma}(p_n, q)),$$

$$\lambda_\sigma((p_1, \ldots, p_n)) = \begin{cases} q_j \text{ if } \min\{k \mid p_k \in F_{q_k,\sigma}\} = j \geq 1, \\ \text{undefined, otherwise.} \end{cases}$$

and E_σ consists of elements of $\prod_{q\in Q} S_{q,\sigma}$ for which λ_σ is defined. The output function λ_σ assigns to a tuple $(p_1, \ldots, p_n) \in E_\sigma$ the vertical state q_j where j is the smallest index such that p_j is a final state of the DFA $H^B_{q_j,\sigma}$. The choice may seem arbitrary, however, the construction works because, since B is a DTA(DFA) the horizontal languages $\delta(q_{j_1}, \sigma)$ and $\delta(q_{j_2}, \sigma)$, $j_1 \neq j_2$, are always disjoint and hence in any tuple $(p_1, \ldots, p_n) \in \prod_{q\in Q} S_{q,\sigma}$ that the computation of $H_\sigma^{B'}$ may actually reach at most one of the components p_i can be a final state of the corresponding horizontal DFA $H^B_{q_i,\sigma}$.

Above we have noted that computations of $H_\sigma^{B'}$ use as states only tuples $(p_1, \ldots, p_n)$ where either, for all $1 \leq j \leq n$, $p_j \in S_{q_j,\sigma} - F_{q_j,\sigma}$, or there exists exactly one $1 \leq j \leq n$, such that $p_j \in F_{q_j,\sigma}$. Eliminating the unnecessary tuples and taking the sum over all $\sigma \in \Sigma$, gives for the total number of horizontal states of B', $\sum_{\sigma\in\Sigma} \text{size}(H_\sigma^{B'})$, the upper bound claimed in the statement of the lemma. The number of vertical states of B' is n. ■

If B has m horizontal states, Lemma 5 gives for the number of horizontal states of B' a worst-case upper bound that is less than 2^m but is not polynomial in m. Next we give a lower bound construction.

Lemma 6. *Let $\Sigma = \{a, b, 0, 1\}$. For any $m \in \mathbb{N}$ and relatively prime numbers $2 \leq k_1 < k_2 < \ldots < k_m$, there exists a tree language L over Σ recognized by a DTA(DFA) B with $\text{size}(B) = [\ m;\ \sum_{i=1}^m k_i + O(m \log m)\ ]$ such that any SDTA recognizing L has at least m vertical states and $\Pi_{i=1}^m k_i$ horizontal states.*

Proof. Let $y_i \in \{0, 1\}^*$ be the binary representation of $i \geq 1$. We define $L = \bigcup_{1\leq i\leq m} a^i((b^{k_i})^* y_i)$.

We define for L a DTA(DFA) $B = (Q, \Sigma, \delta, F)$, where $Q = \{q_1, \ldots, q_m\}$, $F = \{q_1\}$, $\delta(a, q_i) = (b^{k_i})^* \cdot y_i + q_{i+1}$, for $1 \leq i \leq m-1$, and $\delta(a, q_m) = (b^{k_m})^* \cdot y_m$. Note that

the bottom-up computation of B is deterministic because different horizontal languages are marked by distinct binary strings y_i.

Each horizontal language $(b^{k_i})^* \cdot y_i + q_{i+1}$ can be recognized by a DFA with $k_i + \lfloor \log i \rfloor + 3$ states, and in total B has $\sum_{i=1}^{m} k_i + \sum_{i=1}^{m} (\lfloor \log i \rfloor) + 3m$ horizontal states (and m vertical states).

Let $B' = (Q', \Sigma, \delta', F')$ be an arbitrary SDTA recognizing L. By choosing $R = \{a(b^{k_i} y_i) \mid 1 \leq i \leq m\} \cup \{a(b)\}$, Lemma 1 gives $|Q'| \geq m$.

We show that the DFA $H_a^{B'}$, with notations as in (1), defining transitions corresponding to symbol a needs at least $\prod_{i=1}^{m} k_i$ states. Suppose that $H_a^{B'}$ has less than $\prod_{i=1}^{m} k_i$ states. Then there exist $0 \leq j < s < \prod_{i=1}^{m} k_i$ such that $H_a^{B'}$ reaches the same state after reading strings b^j and b^s, respectively. There must exist $1 \leq r \leq m$ such that k_r does not divide $s - j$. Let $z = j + (k_r - j \bmod k_r)$. Since $H_a^{B'}$ reaches the same state on b^j and b^s, it follows that $H_a^{B'}$ reaches the same state also on $b^z \cdot y_r$ and $b^{z+s-j} \cdot y_r$, respectively. This means that $a^{k_r}(b^z y_r)$ is accepted by B' if and only if $a^{k_r}(b^{z+s-j} \cdot y_r)$ is accepted by B', which is a constradiction because k_r divides z and does not divide $z + s - j$. ■

In the above proof, using a mode detailed analysis it could be shown that $H_a^{B'}$ needs $\Omega(m \cdot \log m)$ additional states to process the strings y_i, however, this would not change the worst-case lower bound.

Now we establish that the upper and lower bounds for the DTA(DFA)-to-SDTA conversion are within a multiplicative constant, at least when the sizes of the horizontal DFAs are large compared to the number of vertical states.

Theorem 2. *An arbitrary DTA(DFA) $B = (Q, \Sigma, \delta, F)$ has an equivalent SDTA B' with*

$$\text{size}(B') \leq [\ |Q|;\ \sum_{\sigma \in \Sigma} \prod_{q \in Q} \text{size}(H_{q,\sigma}^{B})\ ], \tag{4}$$

and, for an arbitrary $m \geq 1$ there exists a DTA(DFA) $B = (Q, \Sigma, \delta, F)$ with $|Q| = m$ such that for any equivalent SDTA B' the size of B' has a lower bound within a multiplicative constant of (4).

Proof. The upper bound follows from Lemma 5. We get the lower bound from Lemma 6 by choosing each k_i to be at least $m \cdot \log m$, $i = 1, \ldots, m$. ■

We note that when converting a DTA(DFA) $B = (Q, \Sigma, \delta, F)$ to an equivalent SDTA A, for each $\sigma \in \Sigma$ the horizontal DFA H_σ^A needs at least as many states as a DFA recognizing $L_{B,\sigma} = \bigcup_{q \in Q} \delta(q, \sigma)$. Note that from H_σ^A we obtain a DFA for $L_{B,\sigma}$ simply by ignoring the output function. However, H_σ^A needs to provide more detailed information for a given input string than a DFA simply recognizing $L_{B,\sigma}$, and in fact H_σ^A recognizes the marked union, as formalized below, of the languages $\delta(q, \sigma)$.

We say that a DFA $A = (Q, \Sigma, s_0, F, \gamma)$ equipped with an output function $\lambda : F \to \{1, \ldots, m\}$ recognizes the *marked union* of pairwise disjoint regular languages L_1, ..., L_m,

if $L_i = \{w \in \Sigma^* \mid \lambda(\gamma(s_0, w)) = i\}$, $i = 1, \ldots, m$. The following result establishes that, at least for variable sized alphabets, the state complexity of marked union may be exponentially larger than the state complexity of union.

Proposition 1. *Let $A = (Q, \Sigma, s_0, F, \gamma, \lambda)$ be a DFA with output function $\lambda : F \to \{1, \ldots m\}$ that recognizes the marked union of disjoint languages L_i, $i = 1, \ldots, m$, and let B be the minimal DFA for $\bigcup_{i=1}^{m} L_i$.*

Then $\mathrm{size}(A) \geq \mathrm{size}(B)$*, and there exist disjoint regular languages L_i, $1 \leq i \leq 2^n - 1$, (with $m = 2^n - 1$) over an alphabet of size n, such that* $\mathrm{size}(B) = n + 1$ *and the size of A is at least $2^n - 1$.*

Proof. The inequality $\mathrm{size}(A) \geq \mathrm{size}(B)$ follows from the observation that we obtain B from A simply by ignoring the output function.

Consider an alphabet $\Sigma = \{a_1, a_2, ..., a_n\}$. There are $2^n - 1$ nonempty subsets of Σ. Denote each subset by S_i, $1 \leq i \leq 2^n - 1$. For each $S_i = \{a_{i_1}, \ldots, a_{i_m}\}$, $1 \leq i_1 < i_2 < \cdots < i_m \leq n$, define $x_i = a_{i_1} a_{i_2} \cdots a_{i_m}$. Note that in this way we get one uniquely defined string x_i for each set $\emptyset \neq S_i \subseteq \{a_1, \ldots, a_n\}$.

A DFA recognizing the marked union of $\{x_i\}$, $1 \leq i \leq 2^n - 1$, needs at least as many states as the number of components $\{x_i\}$.

We construct a DFA B with $n + 1$ states for the language $L_0 = \{x_i \mid 1 \leq i \leq 2^n - 1\}$. Choose $B = (Q, \Sigma, 0, Q - \{0\}, \gamma)$ where $Q = \{0, 1, \ldots, n\}$ and the transition function γ is defined by setting for $0 \leq i \leq n$, $1 \leq j \leq n$,

$$\gamma(i, a_j) = \begin{cases} j \text{ if } i < j, \\ \text{undefined, if } i \geq j. \end{cases}$$

Since all states of B except the start state are final, it is easy to verify that B recognizes L_0. The construction in the case $n = 4$ is illustrated in Example 1. ■

Example 1. We consider an example with $\Sigma = \{a, b, c, d\}$. Definte $L_1 = \{a\}$, $L_2 = \{b\}$, $L_3 = \{c\}$, $L_4 = \{d\}$, $L_5 = \{ab\}$, $L_6 = \{ac\}$, $L_7 = \{ad\}$, $L_8 = \{bc\}$, $L_9 = \{bd\}$, $L_{10} = \{cd\}$, $L_{11} = \{abc\}$, $L_{12} = \{abd\}$, $L_{13} = \{acd\}$, $L_{14} = \{bcd\}$, $L_{15} = \{abcd\}$. The DFA that recognizes the union of the languages L_1, ..., L_{15}, is shown in Figure 1.

4 Converting nondeterministic tree automata to deterministic automata

In this section we consider conversions of different variants of nondeterministic automata into equivalent strongly and weakly deterministic automata.

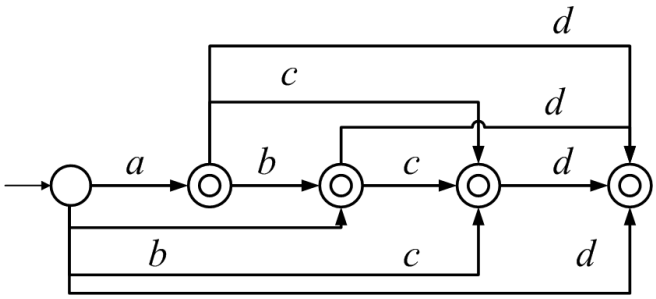

Fig. 1. A DFA recognizing $\bigcup_{i=1}^{15} L_i$

4.1 Converting a nondeterministic automaton to an SDTA

We begin by giving upper bounds.

Lemma 7. *Let $A = (Q, \Sigma, \delta, F)$ be an NTA(NFA) and for $q \in Q$, $\sigma \in \Sigma$ denote* $\mathrm{size}(H^A_{q,\sigma}) = m_{q,\sigma}$.

(i) *We can construct an equivalent SDTA B where*

$$\mathrm{size}(B) \leq [\ 2^{|Q|};\ \sum_{\sigma \in \Sigma} 2^{\left(\sum_{q \in Q} m_{q,\sigma}\right)}\ ]. \tag{5}$$

(ii) *If A is a DTA(NFA), in the upper bound (5) the number of vertical states is at most $|Q|$.*

Proof. First we consider the case (i) where A is an arbitrary NTA(NFA). Denote $Q = \{q_1, \ldots, q_n\}$ and $H^A_{q_i,a} = (C_{a,i}, Q, q^0_{a,i}, F_{a,i}, \gamma_{a,i})$ is the horizontal NFA corresponding to $q_i \in Q$ and $a \in \Sigma$.

We define an SDTA $B = (\mathcal{P}(Q), \Sigma, \eta, F_B)$ where $F_B = \{X \subseteq Q \mid X \cap F \neq \emptyset\}$ and η-transitions corresponding to $a \in \Sigma$ are determined by a DFA with output function:

$$H^B_a = (\mathcal{P}(C_{a,1}) \times \cdots \times \mathcal{P}(C_{a,n}), \mathcal{P}(Q), (\{q^0_{a,1}\}, \ldots, \{q^0_{a,n}\}), E_a, \mu_a, \lambda_a), \tag{6}$$

where μ_a and λ_a are defined below, and E_a consists of all tuples $(Y_1, \ldots, Y_n)$, $Y_i \subseteq C_{a,i}$, $i = 1, \ldots, n$, such that $\lambda_a(Y_1, \ldots, Y_n) \neq \emptyset$. Note that since B uses $\mathcal{P}(Q)$ as the set of states, this is also the input alphabet for H^B_a. For $X \subseteq Q$ and $Y_i \subseteq C_{a,i}$, $i = 1, \ldots, n$,

$$\mu_a((Y_1, \ldots, Y_n), X) = (\bigcup_{x \in X} \gamma_{a,1}(Y_1, x), \ldots, \bigcup_{x \in X} \gamma_{a,n}(Y_n, x)).$$

Here $\gamma_{a,i}(Y_i, x)$ stands for $\bigcup_{z \in Y_i} \gamma_{a,i}(z, x)$. For $Y_i \subseteq C_{a,i}$, $i = 1, \ldots, n$,

$$\lambda_a((Y_1, \ldots, Y_n)) = \{q_i \mid Y_i \cap F_{a,i} \neq \emptyset\}.$$

The computation of H_a^B on a string $w = w_1 \cdots w_k$, $w_j \in \mathcal{P}(Q)$, $j = 1, \ldots, k$, roughly speaking, simulates the computation of each NFA $H_{q_i,a}^A$, $1 \leq i \leq n$, on each string $u = u_1 \cdots u_k \in Q^*$, $u_i \in w_i$, $i = 1, \ldots, k$. With above notations, we say that $u \in Q^*$ is a projection of the string $w \in (\mathcal{P}(Q))^*$. Assume that the computation of H_a^B on w reaches a state $(Y_1, \ldots, Y_n)$, $Y_i \subseteq C_{a,i}$, $i = 1, \ldots, n$. Then λ_a maps $(Y_1, \ldots, Y_n)$ to the set P $(\subseteq Q)$ consisting of exactly those elements $q \in Q$ such that $H_{q,a}^A$ accepts some projection string of the string w. These conditions guarantee that if the nondeterministic computation of A can reach the children of a node v labeled by a in states determined by the string $w \in (\mathcal{P}(Q))^*$, the possible states at node v are exactly the states of P. Note that this property relies strongly on the fact that computations in subtrees corresponding to different children of v are independent. It follows that $L(B) = L(A)$. The SDTA B has 2^n vertical states and $\sum_{a \in \Sigma} 2^{\sum_{i=1}^n |C_{a,i}|}$ horizontal states.

(ii) Now assume that A is a DTA(NFA). Analogously, as in (i) above we define an equivalent SDTA $B = (Q, \Sigma, \eta, F_B)$. Now for the horizontal DFA H_a^B the input alphabet is just Q, however, because the automata $H_{q_i,a}^A$ remain nondeterministic the set of states of H_a^B is, in general, as in (6) and the upper bound for the total number of horizontal states is the same as in (i). ■

We do not require the automata be complete and, naturally, in (5) the number of vertical states of B could be reduced to $2^{|Q|} - 1$. A similar small improvement could be made to the number horizontal states, but it would make the formula look rather complicated.

Also, in Lemma 7 (ii) the upper bound for the number of horizontal states could be slightly reduced using a more detailed analysis, as in the proof of Lemma 5, that takes into account that at most one of the NFAs defining the horizontal languages associated with a fixed input symbol σ can accept simultaneously.

Lemma 7 did not discuss the case where the bottom-up computation is nondeterministic but the horizontal languages are represented in terms of DFAs. We note that for an NTA(DFA) $A = (Q, \Sigma, \delta, F)$ the construction used in the proof of Lemma 7 gives for the size of an equivalent SDTA only the upper bound (5). Although the horizontal languages of A are defined using DFAs, the horizontal languages of the equivalent SDTA B are over the alphabet $\mathcal{P}(Q)$, and this means that the upper bound for the number of horizontal states would not be improved.

Next we state two lower bound results.

Lemma 8. *Let $\Sigma = \{a, b\}$. For any relatively prime numbers $m_1, m_2, ..., m_n$, there exists a tree language L over Σ such that L is recognized by an NTA(DFA) A with* $\mathrm{size}(A) \leq [\ n;\ (\sum_{i=1}^{n} m_i) + 2n - 2\ ]$, *and any SDTA for L needs at least $2^n - 1$ vertical states and $(\prod_{i=1}^{n} m_i) - 1$ horizontal states.*

Proof. We choose $L = \{a^i((b^{m_i})^*) \mid 1 \leq i \leq n\}$.

The tree language L is accepted by an NTA(DFA) $A = (Q, \{a, b\}, \delta, \{q_1\})$, where $Q = \{q_1, q_2, ..., q_n\}$, $\delta(a, q_i) = (b^{m_i})^* + q_{i+1}$, $1 \leq i \leq n-1$, $\delta(a, q_n) = (b^{m_n})^*$. Each horizontal language $\delta(a, q_i)$ can be recognized by a DFA with $m_i + 2$ states, $1 \leq i \leq n-1$, and $\delta(a, q_n)$ is recognized by a DFA with m_n states. Note that since A is an NTA(DFA) there is no requirement that different horizontal languages associated with a would need to be disjoint.

Let $B = (Q', \Sigma, \delta', F')$ be an arbitrary SDTA recognizing L. For $r \subseteq \{1, \ldots, n\}$, define $s_r = a(b^{\Pi_{i \in r} m_i})$. Denote

$$R = \{s_r \mid \emptyset \neq r \subseteq \{1, \ldots, n\}\ \} \cup \{a(b^{(\Pi_{i=1}^n m_i)+1})\}.$$

We show that R satisfies the conditions of Lemma 1. First consider any two distinct nonempty sets $r_1, r_2 \subseteq \{1, \ldots, n\}$. Choose $k \in r_1 - r_2$. The other case where $r_1 - r_2 \neq \emptyset$ is completely symmetric. Now for $t = a^{k-1}(x) \in T_\Sigma[x]$, $t(x \leftarrow s_{r_1}) \in L$ and $t(x \leftarrow s_{r_2}) \notin L$.

Second, for any $\emptyset \neq r$ there exists $t \in T_\Sigma[x]$ such that $t(x \leftarrow s_r) \in L$. On the other hand, for any $t \in T_\Sigma[x]$, $t(x \leftarrow a(b^{(\Pi_{i=1}^n m_i)+1}) \notin L$ because no m_i, $1 \leq i \leq n$, can divide $(\Pi_{i=1}^n m_i) + 1$.

Thus, we have verified that the set R satisfies the conditions of the statement of Lemma 1. Since $|R| = 2^n$ it follows that B needs at least $2^n - 1$ vertical states.

It remains to establish the lower bound for the number of horizontal states of B. Let $K = \Pi_{i=1}^n m_i$ and define $S = \{b^j \mid 1 \leq j \leq K\}$. [4] Consider any distinct integers $1 \leq x < y \leq K$. Since all the m_j's are pairwise relatively prime, there exists $1 \leq i \leq n$ such that m_i does not divide $y - x$. Choose $0 \leq z < m_i$ such that $y + z \equiv 0 \pmod{m_i}$. Also let $t = a^{i-1}[x] \in T_\Sigma[x]$. Now $t(x \leftarrow a(b^{y+z})) \in L$ and $t(x \leftarrow a(b^{x+z})) \notin L$. Note that because m_i divides $y + z$ and m_i does not divide $y - x$, m_i does not divide $x + z$. According to Lemma 2, B needs at least $|S| - 1$ horizontal states. ■

Lemma 9. *For $n \geq 1$, there exists a tree language L_n recognized by a DTA(NFA) A with n vertical and less than $n \log n$ horizontal states such that for any SDTA B for L_n,* size$(B) \geq [\, n;\ 2^n \,]$.

Proof. Let $\Sigma = \{a, b, c, 0, 1\}$ and y_i denotes the binary representation of $i \geq 1$ (without leading zeros). Let L_0 be the language defined by the NFA in Figure 2. Define

$$T = \{c^i(y_i) \mid 1 \leq i \leq n-1\} \cup \{c^n(w) \mid w \in L_0\}.$$

The tree language T is recognized by a DTA(NFA) $A = (Q, \Sigma, \delta, \{q_1\})$, where $Q = \{q_1, q_2, \ldots, q_n\}$, $\delta(c, q_i) = \{y_i, q_{i+1}\}$, $1 \leq i \leq n-1$, $\delta(c, q_n) = L_0$.

Each $\delta(c, q_i)$ can be recognized by an NFA with $\lfloor \log i \rfloor + 2$ states, $1 \leq i \leq n-1$, and $\delta(c, q_n) = L_0$ is recognized by an NFA with n states.

Let $B = \{Q', \Sigma, \delta', F'\}$ be an arbitrary SDTA recognizing T. Define $R = \{c(y_i) \mid 1 \leq i \leq n-1\} \cup \{c(a^{n-1}), c(b)\}$. Let $t \in T_\Sigma[x]$ be arbitrary. For $1 \leq i \leq n-1$, $t(x \leftarrow c(y_i)) \in T$ if

[4] To be consistent with notations of Lemma 2, we view the elements of S as tuples of trees each having one node.

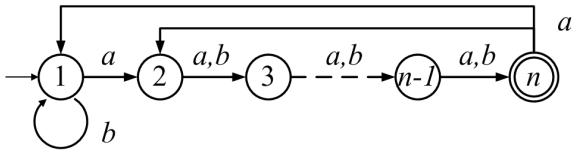

Fig. 2. An NFA for the language L_0.

and only if $t = c^{i-1}(x)$. Also, $t(x \leftarrow c(a^{n-1})) \in T$ if and only if $t = c^{n-1}(x)$ and $t(x \leftarrow c(b))$ is never in T. Thus, R satisfies the conditions of Lemma 1, and it follows that $|Q'| \geq n$.

Let $H_c^B = (S_c, Q', s_c^0, F_c, \gamma_c, \lambda_c)$ be the DFA with output that determines the transitions of B at symbol c.[5] Denote $P = \{q \in Q' \mid B \text{ accepts } c^{n-1}(q)\}$. Define $E = (S_c, Q', s_c^0, \lambda_c^{-1}(P), \gamma_c)$. Now $L(E) = L_0$ and we know by [11] that $\text{size}(E) \geq 2^n$, which means that also H_c^B has at least 2^n states. ■

Using a more detailed analysis of how the horizontal automata process the strings y_i, in the above proof we could slightly improve the lower bound for the number of horizontal states of B.

The lower bounds given by the above two lemmas are far removed from the corresponding upper bounds in Lemma 7. Furthermore, we do not have a worst-case construction for general NTA(NFA)'s that would provably give an essentially better lower bound than the one obtained for NTA(DFA)'s in Lemma 8.

4.2 Converting a nondeterministic automaton to a DTA(DFA)

We begin with a simulation result establishing an upper bound.

Lemma 10. *Let $A = (Q, \Sigma, \delta, F)$ be an NTA(NFA) and for $q \in Q$, $\sigma \in \Sigma$ denote $\text{size}(H_{q,\sigma}^A) = m_{q,\sigma}$.*

(i) *There exists a DTA(DFA) B equivalent to A where*

$$\text{size}(B) \leq [\ 2^{|Q|};\ 2^{|Q|} \cdot (\sum_{\sigma \in \Sigma} 2^{(\sum_{q \in Q} m_{q,\sigma})})\ ]. \tag{7}$$

(ii) *If A is a DTA(NFA), it has an equivalent DTA(DFA) B where*

$$\text{size}(B) \leq [\ |Q|;\ \sum_{q \in Q} \sum_{\sigma \in \Sigma} 2^{m_{q,\sigma}}\ ].$$

Proof. Let $Q = \{q_1, \ldots, q_n\}$ and as usual denote by $H_{q,\sigma}^A = (S_{q,\sigma}, Q, s_{q,\sigma}^0, F_{q,\sigma}, \gamma_{q,\sigma})$ the horizontal DFA corresponding to $q \in Q$ and $\sigma \in \Sigma$.

We use the following notation. For $P = \{q_{i_1}, \ldots, q_{i_m}\} \subseteq Q$, $1 \leq i_1 < i_2 < \cdots < i_m \leq n$, we denote by $I_P = \{i_1, \ldots, i_m\}$ the *index set* of P. We define a DTA(DFA) $B = (\mathcal{P}(Q), \Sigma, \eta, F_B)$

[5] Recall that according to our notational conventions elements of Σ are used also as states of Q'.

where $F_B = \{X \subseteq Q \mid X \cap F \neq \emptyset\}$ and for $P \subseteq Q$ and $\sigma \in \Sigma$, the horizontal language $\eta(P, \sigma)$ is recognized by a DFA

$$H^B_{P,\sigma} = (\mathcal{P}(S_{q_1,\sigma}) \times \cdots \times \mathcal{P}(S_{q_n,\sigma})),\ \mathcal{P}(Q),\ (s^0_{q_1,\sigma}, \ldots, s^0_{q_n,\sigma}), \mathcal{H}_P, \beta_\sigma),$$

where the set of final states is

$$\mathcal{H}_P = \{X_1 \times \cdots \times X_n \mid X_i \subseteq S_{q_i,\sigma}, i = 1, \ldots, m, (\forall i \in I_P)\ \ X_i \cap F_{p_i,\sigma} \neq \emptyset \\ \text{and } (\forall i \in \{1, \ldots, n\} - I_P)\ X_i \cap F_{p_i,\sigma} = \emptyset\},$$

and the transitions are defined by setting for $X_i \subseteq S_{q_i,\sigma}$, $1 \leq i \leq n$, $Y \subseteq Q$,

$$\beta_\sigma(X_1 \times \cdots \times X_n, Y) = (\bigcup_{y \in Y} \gamma_{q_1,\sigma}(X_1, y), \ldots, \bigcup_{y \in Y} \gamma_{q_n,\sigma}(X_m, y)).$$

The above construction and the argument justifying that B correctly simulates A are similar to ones used in the proof of Lemma 7.

We note here just the following. In the above construction $H^B_{P,\sigma}$ simulates also the computation of each NFA $H^A_{q,\sigma}$, $q \in Q - P$. This is necessary,[6] in order to guarantee that the bottom-up computation of B is deterministic, i.e., that all horizontal languages $\eta(P, \sigma)$, $P \subseteq Q$, are pairwise disjoint. In fact, the definition of the DFA $H^B_{P,\sigma}$ depends on P only in the choice of the set of final states $\mathcal{H}_P$.

The above construction of the DFAs $H^B_{P,\sigma}$, $P \subseteq Q$, $\sigma \in \Sigma$, gives the upper bound for the number of horizontal states in (i).

Finally, the upper bound for (ii) follows from the observation that it is sufficient determinize the NFA $H^A_{q,\sigma}$ separately for each $q \in Q$ and $\sigma \in \Sigma$. ■

Roughly speaking, the simulation uses a subset construction for the set of vertical states, and in order to guarantee that the bottom-up computation remains deterministic the DFA for the horizontal language corresponding to $P \subseteq Q$, $\sigma \in \Sigma$, needs to simulate each horizontal NFA of A corresponding to σ. In the case where A is an NTA(DFA) we do not have a significantly better bound than (7), because the horizontal languages of the DTA(DFA) consist of strings of subsets of Q, which means that we again have to simulate multiple computations of each horizontal DFA of A. In the below lower bound construction of Theorem 3 we, in fact, use an NTA(DFA).

We do not have a lower bound that would match the bound of Lemma 10. Recall that strongly deterministic automata can be minimized efficiently and the minimal automaton is unique [3], however, minimal DTA(DFA)'s are, in general, not unique and minimization is intractable [9]. When trying to establish lower bounds for the size of a DTA(DFA) $A = (Q, \Sigma, \delta, F)$ there is the difficulty that by adding more vertical states, and hence more horizontal languages, it may still be possible that the total number of horizontal states is

[6] That is, at least there seems to be no obvious way to avoid simulating all the NFAs $H^A_{q,\sigma}$.

reduced. For example, suppose that A has a horizontal language $\delta(q,\sigma) = (a+b)^*b(a+b)^7$, where the minimal DFA has 256 states.[7] This language can be represented as a disjoint union of 8 languages where the sum of the sizes of the minimal DFAs is only 176. (Details are given in Example 2.) Thus, by replacing the state q by 8 distinct vertical states (that could be equivalent in terms of the bottom-up computation) we could reduce the size of A.

Example 2. We denote the state complexity of a regular language L as $\mathrm{sc}(L)$. Consider the language

$$L_0 = (a+b)^*b(a+b)^7.$$

The minimal DFA for L_0 has 256 states and L_0 can be represented as a disjoint union of languages L_i, $1 \le i \le 8$, that are listed in Table 1 together with the state complexity of each language. From the table we see that $\sum_{i=1}^{8} \mathrm{sc}(L_i) = 176$.

Table 1. State complexity of disjoint union

Language (L_i)	$sc(L_i)$	Language (L_i)	$sc(L_i)$
$L_1 = (a+b)^*bbbb(a+b)^4$	29	$L_2 = (a+b)^*babb(a+b)^4$	23
$L_3 = (a+b)^*bbab(a+b)^4$	23	$L_4 = (a+b)^*bbba(a+b)^4$	19
$L_5 = (a+b)^*baab(a+b)^4$	22	$L_6 = (a+b)^*baba(a+b)^4$	22
$L_7 = (a+b)^*bbaa(a+b)^4$	19	$L_8 = (a+b)^*baaa(a+b)^4$	19

In fact, we do not have a general lower bound condition, analogous to Lemma 2, for the number of horizontal states of DTA(DFA)'s and the below lower bound result relies on an ad hoc proof.

Let $\Sigma = \{a, b\}$. Let $p_1, \ldots, p_n$ be the first n primes. Define the tree language

$$\begin{aligned} T_n = \{a^i(b^k) \mid\ & i \ge 1,\ k \ge 0, \\ & (\exists 1 \le j \le n)[k \equiv 0 \pmod{p_j} \text{ and } i \equiv j \pmod{n}]\}. \end{aligned} \tag{8}$$

Theorem 3. *The tree language T_n can be recognized by an NTA(DFA) A with* $\mathrm{size}(A) = [\, n;\ (\sum_{i=1}^{n} p_i) + 2n \,]$*, and for any DTA(DFA) B recognizing T_n,*

$$\mathrm{size}(B) \ge [\, 2^n - 1;\ (2^n - 1) \cdot \prod_{i=1}^{n} p_i \,].$$

For the proof of Theorem 3 below we need Lemma 11. First we recall some notation concerning unary DFAs. An arbitrary unary DFA can be written as a tuple

$$A = (\{a\}, Q, q_0, F, \delta) \tag{9}$$

[7] Note that $\delta(q,\sigma)$ is a typical example of a language where the NFA-to-DFA size blow-up is large.

where $Q = \{q_0, \ldots, q_{h+k-1}\}$, $h \geq 0$, $k \geq 1$, $F \subseteq Q$ and $\delta(q_i, a) = q_{i+1}$, $0 \leq i < h+k-1$, $\delta(q_{h+k-1}, a) = q_h$. The states $q_0, \ldots, q_{h-1}$ are called the *tail* of A and the states $q_h, \ldots, q_{h+k-1}$ are called the *cycle* of A.

Lemma 11. *Let $p_1, \ldots, p_n$ be the first n primes. Suppose that $\{1, \ldots, n\} = R_1 \cup R_2$ where $R_1 \cap R_2 = \emptyset$. Define*

$$L_0 = \{a^k \mid [(\forall r \in R_1)k \equiv 0 \pmod{p_r}] \text{ and } [(\forall r' \in R_2)k \not\equiv 0 \pmod{p_{r'}}]\}.$$

Assume L_1 is a regular infinite subset of L_0. If A is a DFA recognizing L_1 then the cycle of A has length at least $\Pi_{i=1}^n p_i$.

Proof. We use for A notations as in (9). Now the claim can be written as $k \geq \Pi_{i=1}^n p_i$.

Since L_1 is infinite, the cycle of A has an accepting state q_{h-1+x}, $1 \leq x \leq k$.

First consider an arbitrary $r \in R_1$. Now $a^{h-1+x} \in L_1$ and $a^{h-1+x+k} \in L_1$. Hence $h-1+x \equiv 0 \pmod{p_r}$ and $h-1+x+k \equiv 0 \pmod{p_r}$, which implies that p_r divides k.

Next consider an arbitrary $r' \in R_2$ and for the sake of contradiction assume that $k \not\equiv 0 \pmod{p_{r'}}$. This means that the equation $u \cdot k \equiv 1 \pmod{p_{r'}}$ has a solution u_0 for u. Choose $y \in \{0, \ldots, p_{r'} - 1\}$ such that $h-1+x \equiv y \pmod{p_{r'}}$.

Since $a^{h-1+x} \in L_1$ and k is the length of the cycle of A, it follows that also $a^{h-1+x+(p_{r'}-y)u_0k} \in L_1$. This is a contradiction because, by the choice of u_0 and y, we have $h-1+x+(p_{r'}-y)u_0k \equiv 0 \pmod{p_{r'}}$. ■

Proof of Theorem 3. The tree language T_n can be recognized by an NTA(DFA) $A = (Q, \{a, b\}, \delta, \{q_1\})$, where $Q = \{q_1, q_2, ..., q_n\}$, $\delta(a, q_i) = (b^{p_i})^* + q_{i+1}$, $1 \leq i \leq n-1$, $\delta(a, q_n) = (b^{p_n})^* + q_1$. Each language $\delta(a, q_i)$ can be recognized by a DFA with $p_i + 2$ states, $i = 1, \ldots, n$.

Let $B = (R, \Sigma, r_0, R_F, \delta_B)$ be an arbitrary DTA(DFA) for T_n. Recall that for a Σ-tree t we denote by t^B the state of B reached at the root of t.

We establish a lower bound for the number of vertical and horizontal states of B. It would be fairly easy to establish directly that $|R| \geq 2^n - 1$. We derive this as a consequence of the more general Claim 1 that is useful for a lower bound for the total number of horizontal states.

Define $H_1 = \{a(b^m) \mid m \geq 0\}$. For $t = a(b^m) \in H_1$ we define

$$\text{PRIMES}_t = \{1 \leq j \leq n \mid m \equiv 0 \pmod{p_j}\} \ \ (\subseteq \{1, \ldots, n\}).$$

Then for $S \subseteq \{1, \ldots, n\}$ we define

$$\text{TREES}_S = \{t \in H_1 \mid \text{PRIMES}_t = S\}.$$

TREES_S consists of elements of H_1 where the number of leaves labelled by b is divided by exactly those p_j's where $j \in S$. Furthermore, we define

$$(\text{TREES}_S)_B = \{t^B \mid t \in \text{TREES}_S\}.$$

$(\text{TREES}_S)_B$ consists of states that B reaches at roots of elements of TREES_S.

Claim 1. For any $S_1, S_2 \subseteq \{1, \ldots, n\}$, $S_1 \neq S_2$, we have

$$(\text{TREES}_{S_1})_B \cap (\text{TREES}_{S_2})_B = \emptyset.$$

Proof. For the sake of contradiction assume that

$$r \in (\text{TREES}_{S_1})_B \cap (\text{TREES}_{S_2})_B. \tag{10}$$

Without loss of generality, we can choose $j \in S_1 - S_2$. The other possibility where $S_2 - S_1 \neq \emptyset$ is completely symmetric.

By (10), there exist $t_i = a(b^{m_i}) \in \text{TREES}_{S_i}$, $i = 1, 2$, such that $(t_1)^B = (t_2)^B = r$. By the definition of the sets TREES_{S_i} it follows that $m_1 \equiv 0 \pmod{p_j}$ and $m_2 \not\equiv 0 \pmod{p_j}$.

Choose $u = a^{j-1}(x) \in T_\Sigma[x]$. Since B reaches the same state at the roots of t_1 and t_2, respectively, it follows that

$$u(x \leftarrow t_1) \text{ is accepted by } B \text{ iff } u(x \leftarrow t_2) \text{ is accepted by } B.$$

This is a contradiction because $u(x \leftarrow t_1) \in T_n$ and $u(x \leftarrow t_2) \notin T_n$. ■

As a consequence of Claim 1 it follows that R contains $2^n - 1$ nonempty disjoint sets of states, each of which consists of states reached at the root of elements of TREES_S for some $\emptyset \neq S \subseteq \{1, \ldots, n\}$. In particular, $|R| \geq 2^n - 1$.

For $S \subseteq \{1, \ldots, n\}$ define the unary language

$$\text{UNARY}_S = \{b^m \mid a(b^m) \in \text{TREES}_S\}.$$

By Claim 1,

$$\text{UNARY}_S = \bigcup_{r \in (\text{TREES}_S)_B} \delta_B(r, a). \tag{11}$$

If $(\text{TREES}_S)_B = \{r_0\}$ is a singleton set, the minimal DFA for the horizontal language $\delta_B(r_0, a) = \text{UNARY}_S$ has exactly $\Pi_{i=1}^n p_i$ states.

More generally, UNARY_S can be a finite union of horizontal languages corresponding to states of $(\text{TREES}_S)_B$ as in (11). In this case, one of the languages $\delta_B(r_1, a)$ has to be infinite, and by Lemma 11, the minimal DFA for $\delta_B(r_1, a)$ has a cycle of length at least $\Pi_{i=1}^n p_i$. In all cases, for any nonempty set S ($\subseteq \{1, \ldots, n\}$) the DFAs for the horizontal languages $\delta_B(r, a)$, $r \in (\text{TREES}_S)_B$, have in total at least $\Pi_{i=1}^n p_i$ states. Since S is an arbitrary nonempty subset of $\{1, \ldots, n\}$, we have established the required lower bound for the number of horizontal states of any DTA(DFA) for T_n. ■

Theorem 3 gives a construction where converting an NTA(DFA) to a DTA(DFA) causes an exponential blow-up in the number of vertical states, and additionally the size of each of

the (exponentially many) horizontal DFAs is considerably larger than the original DFA. However, the size blow-up of the horizontal DFAs does not match the upper bound of Lemma 10. In the proof of Theorem 3, roughly speaking, we use particular type of unary horizontal languages in order to be able to (provably) establish that there cannot be a trade-off between the numbers of vertical and horizontal states, and with this type of constructions it seems difficult to approach the worst-case size blow-up of Lemma 10.

5 Conclusion

We have initiated the study of state complexity of conversions between different models of tree automata operating on unranked trees. For the conversion of weakly deterministic automata into strongly deterministic automata, and vice versa, we established lower bounds that are within a multiplicative constant of the corresponding upper bound. However, for the size blow-up of converting nondeterministic automata to (strongly and weakly) deterministic automata the upper and lower bounds remain far apart, and this is a topic for further research.

Since a minimal weakly deterministic automaton need not be unique [9], it is, in general, hard to establish lower bounds for weakly deterministic automata and we do not have tools like Lemma 2 that can be used for strongly deterministic automata. Weakly deterministic automata can have trade-offs between the numbers of vertical and horizontal states, respectively, and it would be useful to establish some upper bounds, for example, for how much the number of horizontal states can be reduced by introducing additional vertical states.

References

1. Brüggemann-Klein, A., Murata, M., Wood, D.: Regular tree and regular hedge languages over unranked alphabets, HKUST Technical report (2001)
2. Carme, J., Niehren, J., Tommasi, M.: Querying unranked trees with stepwise tree automata, Proc. of RTA'04, 105–118 (2004)
3. Cristau, J., Löding, C., Thomas, W.: Deterministic automata on unranked trees, *Proc. of FCT'05,* Lect. Notes Comput. Sci. 3623, Springer, 68–79 (2005)
4. Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M.: *Tree Automata Techniques and Applications,* electronic book available at `www.grappa.univ-lille3.fr/tata`, (2002)
5. Gécseg, F., Steinby, M.: Tree languages, in: Rozenberg, G., Salomaa, A. (eds.), *Handbook of Formal Languages,* vol. III, Springer, Berlin, pp. 1–68 (1997)
6. Holzer, M., Kutrib, M.: Descriptional and computational complexity of finite automata, *Proc. of LATA'09,* Lect. Notes Comput. Sci. 5457, Springer, 23–42 (2009)
7. Hopcroft, J.E., Ullman, J.D.: *Introduction to Automata Theory, Languages, and Computation,* Addison Wesley (1979)
8. Liu, G., Martin-Vide, C., Salomaa, A., Yu, S.: State complexity of basic language operations combined with reversal, *Inform. Comput.* 206, 1178–1186 (2008)
9. Martens, W., Niehren, J.: On the minimization of XML schemas and tree automata for unranked trees, *J. Comput. System Sci.* 73, 550–583 (2007)
10. Milo, T., Suciu, D., Vianu, V.: Typechecking for XML transformers, *J. Comput. System Sci.* 66, 66–97 (2003)

11. Moore, F. R.: 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, 1211–1214 (1971)
12. Salomaa, K.: Descriptional complexity of nondeterministic finite automata, *Proc. DLT'07,* Lect. Notes Comput. Sci. 4588, Springer, 31–35 (2007)
13. Schwentick, T.: Automata for XML — a survey, *J. Comput. System Sci.* 73, 289–315 (2007)
14. Yu, S.: Regular languages, in: Rozenberg, G., Salomaa, A. (eds.), *Handbook of Formal Languages,* vol. I, Springer, Berlin, pp. 41–110 (1997)