Technical Report No. 2011-577
State Complexity of Star and Quotient Operation for Unranked Tree Automata

Xiaoxue Piao and Kai Salomaa

School of Computing, Queen's University,
KINGSTON, Ontario K7L3N6, Canada
{piao, ksalomaa}@cs.queensu.ca

Abstract. We consider the state complexity of extensions of the Kleene star and quotient operations to unranked tree languages. Due to the nature of the tree structure, there are two distinct ways to define the star operation for trees, we call these operations, respectively, bottom-up and top-down star. We show that $(n + \frac{3}{2})2^{n-1}$ states are sufficient and necessary in the worst case to recognize the bottom-up star of a tree language recognized by an n-state deterministic unranked tree automaton. The bound is of a different order than the known state complexity result $\frac{3}{2} \cdot 2^{n-1}$ for the Kleene star operation for automata on strings. On the other hand, for the top-down star we obtain a tight state complexity bound that coincides with the corresponding result for automata on strings.
The bottom-quotient and top-quotient operations are extensions of the left and right quotient to trees. We establish tight state complexity bounds for both variants of quotient. The precise worst-case state complexity of bottom-quotient is shown to be $(n+1)2^n - 1$, which differs by the multiplicative factor $n + 1$ from the corresponding result $2^n - 1$ for ordinary finite automata.

Keywords: unranked trees, deterministic tree automata, Kleene star, quotient operation, operational state complexity.

1 Introduction

XML plays an important role in data representation and exchange on the web [2]. Since its arrival, many theories in formal languages and automata have been used in XML applications, which include tree grammars and automata [9, 8, 13, 12]. As one of the automaton models for XML, we consider unranked tree automata in this paper. For other models such as nested word automata and stepwise automata, please refer to [1, 11, 3] and the references listed there.

In recent years, a lot of work has been done on the descriptional complexity of finite automata and related structures [5–7, 14, 15]. Operational state complexity studies how the size of an automaton changes under regularity preserving operations. While the corresponding results for string languages are well known [14, 16], very few results have been obtained for tree automata. While the state complexity results for tree automata operating on ranked trees are often similar to the corresponding results on regular string automata, the situation becomes essentially different for automata operating on unranked trees.

In our previous paper [10], we have studied union, intersection and concatenation of deterministic unranked tree automata. In this paper, we continue the work and study the state complexity of star operations on deterministic unranked tree automata. We find that due to the nature of the tree structure, there are two essentially different ways to compute the star of tree languages. We name them top-down star operation and bottom-up star operation.

The top-down star operation is defined by building the concatenation of trees "top-down". Thus, we define the kth top-down power of a tree language T to consist of all trees that are obtained from some tree in the $(k-1)$th power of T by replacing some leaf by a tree $t \in T$. Since there is no restriction on where the substitution occurs, this means that the kth power of T includes, among others, all trees obtained from a tree in T where $k-1$ leaves have been replaced by some tree of T.

For the bottom-up star operation we define the kth bottom-up power of T by replacing some leaf of a tree $t \in T$ by a tree in the $(k-1)$th power of T. Thus, in some sense the trees are concatenated sequentially for the bottom-up star, and, in parallel for the top-down star. From this it might seem that the bottom-up star is more similar to the classical star operation than the top-down star. However, at least in terms of state complexity, just the opposite turns out to be the case.

We find that for a tree language T recognized by an n vertical state deterministic unranked tree automaton, $\frac{3}{2} \cdot 2^{n-1}$ is a tight upper bound on the number of vertical states for the top-down star operation which is the same as that of the star operation on ordinary strings, and $(n + \frac{3}{2})2^{n-1}$ vertical states are sufficient and necessary in the worst case for a deterministic unranked tree automaton to recognize the bottom-up star of T, which is of a different order than the known result $\frac{3}{2} \cdot 2^{n-1}$ on strings. The factor n is necessary here due to the restriction that the kth power of T consists of trees obtained from some $t \in T$ by replacing one leaf by a tree in the $(k-1)$th power of T. This means the computation must

guarantee that each concatenated tree has at most one leaf substituted by another tree.

We define top-quotient and bottom-quotient operations on trees which correspond, respectively, to right and left quotient when dealing with automata that process the input tree from the leaves to the root. We investigate the state complexity of these two operations on deterministic unranked tree automata. For a tree language T recognized by an n vertical state deterministic unranked tree automaton, n vertical states are necessary and sufficient for any deterministic unranked tree automaton to recognize the top-quotient of T with respect to an arbitrary tree language. However, for the bottom-quotient operation, the tight state complexity bound is $(n+1)2^n - 1$ which is of a different order than the state complexity of left-quotient for automata operating on strings. Recall that the state complexity of left-quotient is $2^n - 1$ [14]. The factor $n+1$ is necessary here because the automaton has to be sure that the concatenation takes place in only one branch of the tree.

The paper is organized as follows. Definitions of top-down and bottom-up star operation, top-quotient and bottom-quotient on trees and other notations are given in Section 2. We prove the tight upper bounds on the number of vertical states for bottom-up and top-down star operations in Section 3. We give tight bounds for quotient operations in Section 4. All proofs omitted due to the length restriction can be found in the appendix at the end of the paper.

2 Preliminaries

Here we briefly recall some notations. A general reference on tree languages and tree automata is [4].

Let $\mathbb{N}$ be the set of non-negative integers. A *tree domain* D is a finite set of elements in $\mathbb{N}^*$ with the following two properties: (i) If $w \in D$ and u is a prefix of w then $u \in D$. (ii) If $ui \in D$, $i \in \mathbb{N}$ and $j < i$ then $uj \in D$. The nodes in an unranked tree t can be denoted by a tree domain $dom(t)$, and t is a mapping from $dom(t)$ to the set of labels Σ. The set of Σ-labeled trees is $TREES_\Sigma$.

We denote a tree $t = b(a_1, \ldots, a_n)$, whose root is labeled by b and leaves are labeled by $a_1, \ldots, a_n$, simply as $b(a_1 \ldots a_n)$. When $a_1 = \ldots = a_n = a$, write $t = b(a^n)$. By a slight abuse of notation, for a unary tree $t = a_1(a_2(\ldots(a_n)\ldots))$, we write $t = a_1 a_2 \ldots a_n$ for abbreviation. When $a_1 = \ldots = a_n = a$, we write $t = a^n$ for short. (In each case it should be

clear from the context whether a^n refers to a sequence of leaves or to a unary tree.)

The set of all Σ-trees where exactly one leaf is labeled by a special symbol X ($X \notin \Sigma$) is $TREES_\Sigma[X]$. For $t \in TREES_\Sigma[X]$ and $t' \in TREES_\Sigma$, $t(X \leftarrow t')$ denotes the tree obtained from t by replacing the unique occurrence of the variable X by t'.

We define tree concatenation as an operation where *one leaf* of a tree t is replaced by another tree t'. Furthermore, we can restrict the places where the substitution can occur depending on the leaf label. For $t, t' \in TREES_\Sigma$ and $b \in \Sigma$, we denote by $t \cdot_b t'$ the set of trees that are obtained from t by replacing one leaf labeled by b by the tree t'. The b-concatenation operation is extended in the natural way to sets of trees L_1, L_2:

$$L_1 \cdot_b L_2 = \bigcup_{t \in L_1, t' \in L_2} t \cdot_b t'.$$

Definition 1. *The top-down b-star operation of a tree language L for $b \in \Sigma$, denoted as $[L]^*_{(b)}$, is an infinite union $\bigcup_{i \geq 0} [L]^i_{(b)}$, where $[L]^0_{(b)} = \{b\}$, $[L]^1_{(b)} = L$, and $[L]^{i+1}_{(b)} = ([L]^i_{(b)}) \cdot_b L$, for $i \geq 1$.*

Definition 2. *The bottom-up b-star operation of a tree language L for $b \in \Sigma$, denoted as $L^*_{(b)}$, is an infinite union $\bigcup_{i \geq 0} L^i_{(b)}$, where $L^0_{(b)} = \{b\}$, $L^1_{(b)} = L$, and $L^{i+1}_{(b)} = L \cdot_b (L^i_{(b)})$, for $i \geq 1$.*

For example, if $L = \{a(b, b)\}$, tree $a(a(b, b), a(b, b)) \in [L]^3_{(b)}$ and $a(a(b, b), a(b, b)) \notin L^3_{(b)}$. When $b \in \Sigma$ is understood from the context, we simply call the operation top-down (respectively bottom-up) star operation and write $[L]^*$ (respectively L^*) in place of $[L]^*_{(b)}$ (respectively $L^*_{(b)}$). The b-concatenation is not associative. The top-down and bottom-up star operations represent exactly the left- and right-most bracketing of this non-associative b-concatenation operation. Consider the concatenation of the sequence $t_0, t_1, \ldots, t_{i+1} \in L$. By Definition 1, t_{i+1} can replace any leaf node in t_j, $j \leq i$. However, by Definition 2, t_{i+1} can only replace any leaf node in t_i. Figure 1(a) shows one possible resulting tree constructed by using Definition 1, and Figure 1(b) shows a resulting tree obtained by Definition 2.

Next we define the quotient operations considered in this paper.

Definition 3. *Let $b \in \Sigma$. The b-top-quotient of a tree language T with respect to a tree language T' is defined as:*

$$T' \top_b T = \{t \mid \exists t' \in T', t' \cdot_b t \in T\}.$$

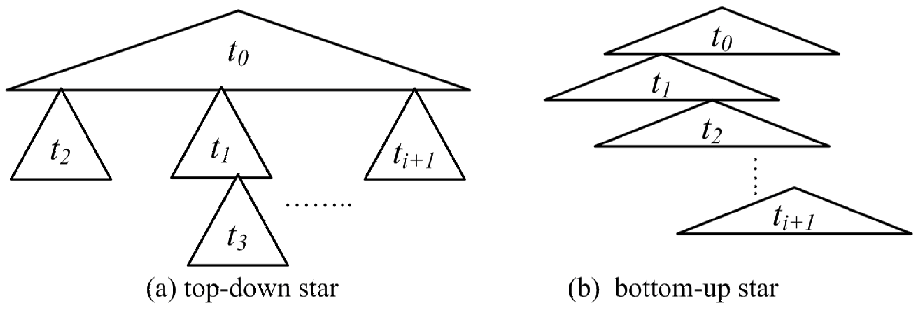

Fig. 1. Two resulting trees from two definitions

Definition 4. *Let $b \in \Sigma$. The b-bottom-quotient of a tree language T with respect to a tree language T' is defined as:*

$$T \perp_b T' = \{t \mid \exists t' \in T', t \cdot_b t' \in T\}.$$

When considering computations that process a tree in the bottom-up direction, the top-quotient can be viewed as an extension of right-quotient from strings to trees, and similarly, the bottom-quotient extends the left-quotient operation from strings to trees. When $b \in \Sigma$ is understood from the context, we simply call the operations top-quotient and bottom-quotient and write $T' \top T$ (respectively $T \perp T'$) in place of $T' \top_b T$ (respectively $T \perp_b T'$).

A deterministic unranked tree automaton (DTA(DFA))[1] 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 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 and for each label σ and every two states $q_1 \neq q_2$, $\delta(q_1, \sigma) \cap \delta(q_2, \sigma) = \emptyset$. The language $\delta(q, \sigma)$ is called the *horizontal language* associated with q and σ and it is specified by a DFA $H^A_{q,\sigma}$.

The states in Q are called *vertical states*. The DFAs recognizing the horizontal languages are called *horizontal DFAs* and their states are called horizontal states. For a tree t, $t^A \in Q$ denotes the state assigned to the root of t by A in the computation. The set of all Σ-trees where some leaf nodes are labeled by elements of Q is $TREES_\Sigma(Q)$.

To simplify the constructions, we assume that the DFAs representing the horizontal languages are complete.

[1] Following the terminology from [10, 12] we call a deterministic bottom-up tree automaton where the horizontal languages are represented by DFAs a DTA(DFA).

3 State complexity of star operation

In this section, we prove tight upper bounds on the numbers of vertical states for both top-down and bottom-up star operations on deterministic unranked tree automata. We present upper bounds on both star operations below, and state the corresponding matching lower bound examples in Section 3.2.

3.1 Upper bounds

First we state the upper bound for the bottom-up star operation.

Lemma 1. *Let $A = (Q, \Sigma, \delta, F)$ be an arbitrary DTA(DFA), and $H^A_{\sigma,q} = (C_{\sigma,q}, Q, \mu_{\sigma,q}, c_{\sigma,q,0}, U_{\sigma,q})$ be the DFA representing the horizontal language associated with σ and q.*

We can construct a DTA(DFA) B recognizing the language $L(A)^*_{(b)}$, $b \in \Sigma$ (bottom-up b-star operation) with at most

$$(|Q| + \frac{3}{2})2^{|Q|-1}$$

vertical states.

Proof. Let $Q = \{s_1, s_2, \ldots, s_n\}$ and $0 \in Q$ be the state assigned to the leaf nodes labeled by b. Choose $B = (P, \Sigma, \lambda, E)$, where $P = P_1 \cup P_2 \cup P_3$,

- $P_1 = \{(\{q\} \cup p, q) \mid p \in \mathcal{P}(Q - \{q\}), q \in Q - F\}$,
- $P_2 = \{(\{q, 0\} \cup p, q) \mid p \in \mathcal{P}(Q - \{q, 0\}), q \in F\}$,
- $P_3 = \{(p, dead) \mid p \in \mathcal{P}(Q)\}$.

State (p, q) in P is final if q is in F or there exists state q' in p such that q' is in F. In the following, we assume that $\{0\} \cap F \neq F$, which means there is a final state different from 0.

For states $(p \cup \{q\}, q) \in P_1$ and states $(p \cup \{q, 0\}, q) \in P_2$, let $p = \{q_1, \ldots, q_m\}$, $0 \leq m < n$. The transition function $\lambda(\sigma, (p, q))$ is defined by DFA $G_{\sigma,(p,q)} = (D_{\sigma,(p,q)}, P, \gamma_{\sigma,(p,q)}, d_{\sigma,(p,q),0}, V_{\sigma,(p,q)})$. The set of states is $D_{\sigma,(p,q)} = ((\mathcal{P}(C_{\sigma,q_1}) \times \ldots \times \mathcal{P}(C_{\sigma,q_m})) \times (C_{\sigma,q_1} \times \ldots \times C_{\sigma,q_m}) \times C_{\sigma,q}$. The initial state is $d_{\sigma,(p,q),0} = ((\{c_{\sigma,q_1,0}\}, \ldots, \{c_{\sigma,q_m,0}\}), (c_{\sigma,q_1,0}, \ldots, c_{\sigma,q_m,0}), c_{\sigma,q,0})$. The set of final states $V_{\sigma,(p,q)}$ consists of all the states $v = ((C_1, \ldots, C_m), (u_1, u_2, \ldots, u_m), u)$ where for every $1 \leq i \leq m$, $C_i \cap U_{\sigma,q_i} \neq \emptyset$, $u \in U_{\sigma,q}$.

The transition function $\gamma_{\sigma,(p,q)}$ is defined as follows: for any input $(p',q') = (\{j_1, j_2, \ldots, j_i\}, q')$ and any state $d = ((C_1, \ldots, C_m), (c_1, c_2, \ldots, c_m), c)$ in $D_{\sigma,(p,q)}$,

$$\gamma_{\sigma,(p,q)}((p',q'),d) =$$
$$((\bigcup_{c\in C_1} \mu_{\sigma,q_1}(q',c) \cup \bigcup_{l=1}^{i} \mu_{\sigma,q_1}(j_l,c_1), \ldots, \bigcup_{c\in C_m} \mu_{\sigma,q_m}(q',c) \cup \bigcup_{l=1}^{i} \mu_{\sigma,q_m}(j_l,c_m)),$$
$$(\mu_{\sigma,q_1}(q',c_1), \ldots, \mu_{\sigma,q_m}(q',c_m)), \mu_{\sigma,q}(q',c))$$

The γ-transition in the horizontal DFA $G_{\sigma,(p,q)}$ of B is defined using the μ-transition in the horizontal DFAs in A. However, since the input of $G_{\sigma,(p,q)}$ is a sequence of states of B, e.g. $(p_1,q_1),(p_2,dead),\ldots,(p_m,q_m)$, *dead* is also a possible input for the μ-transition. Since *dead* is not in the alphabet of the horizontal DFAs of A, which means that μ is undefined for input *dead*, we define the μ-transition in the horizontal DFAs in A such that any transition labeled by *dead* goes to the sink state of the DFAs.

For states in $(p, dead) \in P_3$, let $p = \{q_1, \ldots, q_m\}$, $m \leq n$. The transition function $\lambda(\sigma,(p,dead))$ is defined by the DFA $G_{\sigma,(p,dead)} = (D_{\sigma,(p,dead)}, P, \gamma_{\sigma,(p,dead)}, d_{\sigma,(p,dead),0}, V_{\sigma,(p,dead)})$. The set of states is $D_{\sigma,(p,dead)} = ((\mathcal{P}(C_{\sigma,q_1}) \times \ldots \times \mathcal{P}(C_{\sigma,q_m})) \times ((C_{\sigma,q_1}) \times \ldots \times C_{\sigma,q_m}) \times (C_{\sigma,s_1}, \ldots, C_{\sigma,s_n})$. The initial state is $d_{\sigma,(p,dead),0} = ((\{c_{\sigma,q_1,0}\}, \ldots, \{c_{\sigma,q_m,0}\}), (c_{\sigma,q_1,0}, \ldots, c_{\sigma,q_m,0}), (c_{\sigma,s_1,0}, \ldots, c_{\sigma,s_n,0}))$. The set of final states $V_{\sigma,(p,dead)}$ consists of all the states $v = ((C_1, \ldots, C_m), (u_1, u_2, \ldots, u_m), (v_1, \ldots, v_n))$ where for every $1 \leq i \leq m$, $C_i \cap U_{\sigma,q_i} \neq \emptyset$, and there does not exist any v_i that $v_i \in \bigcup_{s\in Q} U_{\sigma,s}$.

The transition function $\gamma_{\sigma,(p,dead)}$ is defined as follows: for any input $(p',q') = (\{j_1, j_2, \ldots, j_i\}, q')$ and any state $d = ((C_1, \ldots, C_m), (c_1, c_2, \ldots, c_m), (c^1, \ldots, c^n))$ in $D_{\sigma,(p,q)}$,

$$\gamma_{\sigma,(p,dead)}((p',q'),d) =$$
$$((\bigcup_{c\in C_1} \mu_{\sigma,q_1}(q',c) \cup \bigcup_{l=1}^{i} \mu_{\sigma,q_1}(j_l,c_1), \ldots, \bigcup_{c\in C_m} \mu_{\sigma,q_m}(q',c) \cup \bigcup_{l=1}^{i} \mu_{\sigma,q_m}(j_l,c_m)),$$
$$(\mu_{\sigma,q_1}(q',c_1), \ldots, \mu_{\sigma,q_m}(q',c_m)), (\mu_{\sigma,q_1}(q',c^1), \ldots, \mu_{\sigma,q_n}(q',c^n)))$$

The vertical states in B consist of two components. The first component simulates the situation where the concatenation occurs, and the second component records the computation where there is no concatenation. At a subtree $\sigma(t_1, t_2, \ldots, t_m)$, where states $(p_1,q_1),(p_2,q_2),\ldots,(p_m,q_m)$,

$p_i \subseteq Q, 1 \leq i \leq m$ are assigned to the roots of $t_1, t_2, \ldots, t_m$, respectively, state (p, q) is assigned to the root labeled by σ provided that the sequence of states $q_1 q_2 \ldots q_m$ is accepted by $H^A_{\sigma,q}$, and p consists of all states that are obtained by A by in the child nodes "taking" one state in p_i and using q_j, for all $j \neq i$ for all choices of i. The state $(p, dead)$, $p \neq Q$ is assigned to the node labeled by σ if none of the DFAs $H_{\sigma,q}$ in A accepts the sequence $q_1 q_2 \ldots q_m$. When the vertical computation produces a state with the second component in F, the computation adds 0 to the first component of the state.

For states (p, q) in P_1, we always have $q \in p$. This is done because for any state (p, q) where $q \notin p$ there is always an equivalent state $(p \cup \{q\}, q)$. Recall that a state (p, q) is final if $q \in F$ or $p \cap F \neq \emptyset$. Thus, after B reading some tree, B reaches a final state from the state (p, q) if and only if the state $(p \cup \{q\}, q)$ reaches a final state.

According to the construction, $|P_1| = (|Q| - |F|)2^{|Q|-1}$, $|P_2| = |F|2^{|Q|-2}$, $|P_3| = 2^{|Q|}$. We have $|P| = (|Q| - \frac{|F|}{2} + 2)2^{|Q|-1}$. The worst case is when $|F| = 1$. Then the number of vertical states of B is at most $(|Q| + \frac{3}{2})2^{|Q|-1}$. ■

The lemma below states an upper bound for the top-down star operation.

Lemma 2. *Let $A = (Q, \Sigma, \delta, F)$ be an arbitrary DTA(DFA). We can construct a DTA(DFA) B accepting language $[L(A)]^*_{(b)}$ (top-down b-star operation) with at most*

$$2^{|Q|-|F|-1} + 2^{|Q|-|F|}$$

vertical states.

Proof. Since Definition 1 has no restriction on the place where a leaf can be replaced, the vertical state of B needs only one component P, $P \subseteq Q$. Initially, P contains only the state p_b assigned to the leaf node b. B simulates the computation of A. Whenever a final state is produced in P, p_b is added to P. B accepts the input if P contains some state p such that $p \in F$. ■

We are going to state corresponding matching lower bounds in the next subsection.

3.2 Lower bounds

First, we present a lower bound for the bottom-up star operation as follows. In Figure 2 we show the DFA A we get from the one used in Theorem 5 of [16] by adding symbol c.

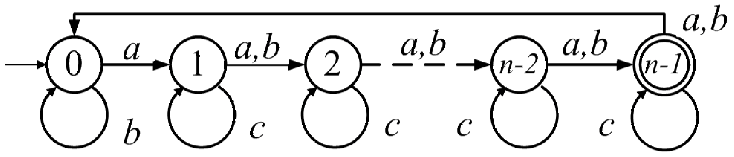

Fig. 2. The DFA A used to define the tree language T

Based on the DFA A we define the tree language T used in our lower bound construction. The tree language T consists of Σ-labeled trees t, $\Sigma = \{a, b, c\}$, where:

1. All leaves are labeled by a and if a node u has a child that is a leaf, then all the children of u are leaves.
2. A accepts the string of symbols labeling a path from any node of height one [2] to the root.
3. The following holds for any $u \in \mathrm{dom}(t)$ and any nodes v_1 and v_2 of height one below u. If w_i is the string of symbols labeling the path from v_i to u, $i = 1, 2$, then A reaches the same state after reading strings w_1 and w_2.

The computation starts from nodes of height one. This is done because when the concatenation takes place, the leaf node is substituted by a tree, and it will not appear in the resulting tree.

T can be recognized by a DTA(DFA) $M = (Q, \{a, b, c\}, \delta, F)$ where $Q = \{0, 1, \ldots, n-1\}$ and $F = \{n-1\}$. The transition function is defined as:

(1) $\delta(0, a) = (n-1)^*$,
(2) $\delta(i, a) = (i-1)^+$ for $1 \leq i \leq n-1$,
(3) $\delta(j, b) = (j-1)^+$ for $2 \leq j \leq n-1$,
(4) $\delta(0, b) = \{n-1, 0\}^+$,
(5) $\delta(i, c) = i^+$ for $1 \leq i \leq n-1$.

[2] We define here that the height of a tree that contains only a root node is 0

A DTA(DFA) C recognizing the tree language T^* is obtained from M using the construction given in Lemma 1. The vertical states in C are of the following form

$$(p, q), 0 \leq q \leq n, p \subseteq \{0, 1, \ldots, n-1\}, \tag{1}$$

where

- if $0 \leq q \leq n-1$, then $q \in p$,
- if $q = n-1$ then $0 \in p$,
- $q = n$ denotes the *dead* state in the construction.

Lemma 3. *All states in (1) are reachable.*

Proof. For a unary tree $t = a_1(a_2(\ldots a_m(a)\ldots))$, we denote $word(t) = a_m a_{m-1} \ldots a_1 \in \Sigma^*$. In the following discussion all numbers that are used to represent elements of Q are interpreted modulo n. That is, y is interpreted as the element of $\{0, 1, \ldots, n-1\}$ congruent to y modulo n.

From state $(\{0\}, 0)$ state $(\{0, n-1\}, n-1)$ can be reached by reading $n-1$ $a's$. State $(\{1, 0\}, 0)$ can be reached by reading a. State $(\{2, 0\}, 0)$ can be reached by reading b. State $(\{0, 1, n-1\}, n-1)$ can be reached by reading $n-1$ $a's$. Similarly, state $(\{0, 1, 2, n-1\}, n-1)$ can be reached by reading sequence ab followed by $n-1$ $a's$.

Assume that state $(\{0, 1, 2, \ldots, k, n-1\}, n-1)$, $0 \leq k < n-2$ is reachable. We want to show that $(\{0, 1, 2, \ldots, k+1, n-1\}, n-1)$ is also reachable. From state $(\{0, 1, 2, \ldots, k, n-1\}, n-1)$, state $(\{1, 2, \ldots, k+1, 0\}, 0)$ is reachable by reading a. Since $0 \leq k < n-2$, $k+1 < n-1$. State $(S_1, 0) = (\{2, 3, \ldots, k+2, 0\}, 0)$ is reachable by reading b. Since $0 \leq k < n-2$, $k+2 < n$ which means $(k+2) \bmod n \neq 0$. (If $(k+2) \bmod n = 0$, the size of S_1 will decrease by 1.) After reading $n-1$ $a's$, state $(\{1, 2, \ldots, k+1, n-1, 0\}, n-1)$ is reached.

We have proved that state $(\{0, 1, \ldots, n-2, n-1\}, n-1)$ is reachable. After reading $i+1$ $a's$, state $(\{0, 1, \ldots, n-2, n-1\}, i \bmod n)$ is reached.

Assume all states (S, j), $|S| \geq k+1$, $k < n$ and $0 \leq j \leq n-1$ in (1) are reachable. We will inductively show that all states where $|S| = k$ are reachable. Let (S, s_i) where $S = \{s_1, s_2, \ldots, s_k\}$, $1 \leq i \leq k$ and $0 \leq s_1 < s_2 < \ldots < s_k \leq n-1$ be an arbitrary state where $|S| = k$.

First consider the case where $s_{i+1} - s_i \geq 2$ and $s_i < n-1$. According to the inductive assumption, state $(\{0, n-1\} \cup S_1, 0)$ where $S_1 = \{s_{i+1} - s_i - 1, s_{i+2} - s_i - 1, \ldots, s_k - s_i - 1, n + s_1 - s_i - 1, \ldots, n + s_{i-1} - s_i - 1\}$ is reachable. Since $0 \leq s_1 < s_2 < \ldots < s_k \leq n-1$ and $s_i - s_{i-1} \geq 2$, $n-1 \notin S_1$ and $0 \notin S_1$. After reading b, state $(\{0\} \cup S_2, 0)$ where $S_2 =$

$\{s_{i+1}-s_i, s_{i+2}-s_i, \ldots, s_k-s_i, n+s_1-s_i, \ldots, n+s_{i-1}-s_i\}$ is reached. Since $0 \leq s_1 < s_2 < \ldots < s_k \leq n-1$, $0 \notin S_2$. State $(\{s_i, s_{i+1}, s_{i+2}, \ldots, s_k, n+s_1, \ldots, n+s_{i-1}\}, s_i)$ is reached after reading s_i $a's$. This means we have reached the state (S, s_i), where $S = \{s_1, s_2, \ldots, s_k\}$.

Next consider the case when $s_{i+1} = s_i + 1$ and $s_i < n-1$. Since $|S| = k < n$, there exists $s_j, s_{j+1} \in S$ such that $s_{j+1} - s_j \geq 2$. According to the inductive assumption, state $(\{0, n-1\} \cup S_1, n+s_i-s_j-1)$ where $S_1 = \{s_{j+1}-s_j-1, s_{j+2}-s_j-1, \ldots, s_k-s_j-1, n+s_1-s_j-1, n+s_2-s_j-1, \ldots, n+s_i-s_j-1, n+s_i-s_j, \ldots, n+s_{j-1}-s_j-1\}$ is reachable. After reading b, $(\{0\} \cup S_2, n+s_i-s_j)$ where $S_2 = \{s_{j+1}-s_j, s_{j+2}-s_j, \ldots, s_k-s_j, n+s_1-s_j, n+s_2-s_j, \ldots, n+s_i-s_j, n+s_i-s_j+1, \ldots, n+s_{j-1}-s_j\}$ is reachable. Since $0 \leq s_1 < s_2 < \ldots < s_k \leq n-1$, $0 \notin S_2$. State $(\{s_j, s_{j+1}, s_{j+2}, \ldots, s_k, n+s_1, n+s_2, \ldots, n+s_i, n+s_i+1, \ldots, n+s_{j-1}\}, n+s_i)$ is reached after reading s_j $a's$. This means we have reached the state (S, s_i), where $S = \{s_1, s_2, \ldots, s_k\}$.

Now consider the case when $s_i = n-1$. When $s_i = n-1$, $0 \in S$. Since $0 \leq s_1 < s_2 < \ldots < s_k \leq n-1$, we have $s_i = s_k = n-1$ and $s_1 = 0$. Since $k < n$, there always exists $s_j, s_{j+1} \in S$ such that $s_{j+1} - s_j \geq 2$. According to the inductive assumption, state $(\{s_{j+1}-s_j-1, s_{j+2}-s_j-1, \ldots, s_{k-1}-s_j-1, n-1-s_j-1, n+0-s_j-1, n+s_2-s_j-1, \ldots, n+s_{j-1}-s_j-1, n-1, 0\}, n-1-s_j-1)$ is reachable. State $(\{s_{j+1}-s_j, s_{j+2}-s_j, \ldots, s_{k-1}-s_j, n-1-s_j, n+0-s_j, n+s_2-s_j, \ldots, n+s_{j-1}-s_j, 0\}, n-1-s_j)$ is reached after reading b. After reading s_j $a's$, state $(\{s_{j+1}, s_{j+2}, \ldots, s_{k-1}, n-1, n+0, n+s_2, \ldots, n+s_{j-1}, s_j\}, n-1)$ is reached. This means we have reached the state $(S, n-1)$ where $S = \{0, s_2, \ldots, s_{k-1}, n-1\}$.

Thus, we have shown all states (S, x), $S \subseteq \{0 \leq j \leq n-1\}$, $0 \leq x \leq n-1$ in (1) are reachable. Next we will show that states (S, n), $S \subset \{0, 1, \ldots, n-1\}$ are reachable.

We have already shown that $(\{0, 1, \ldots, n-1\}, 0)$ is reachable. State $(\{1, \ldots, n-1\}, n)$ is reachable by reading c. State $(\{0, 1, \ldots, n-1\} - \{i\}, n)$, $1 \leq i \leq n-1$ is reachable after reading i $a's$. Assume all states (S, n), $n > |S| \geq k+1$, $k < n-1$ are reachable. We will inductively show that all states where $|S| = k$ are reachable. Let (S, n) where $S = \{s_1, s_2, \ldots, s_k\}$, $0 \leq s_1 < s_2 < \ldots < s_k \leq n-1$ be an arbitrary state where $|S| = k$. According to the inductive assumption, state $(S \cup \{0\}, n)$ is reachable. State (S, n) is reachable by reading c.

That is all the states in (1) except state $(\{0, 1, \ldots, n-1\}, n)$ are reachable.

We show that state $(\{0,1,\ldots,n-1\},n)$ is reachable by introducing a new symbol d. Let T be the tree language defined above Lemma 3 and M is the deterministic tree automaton constructed for T above Lemma 3. Let d be a new symbol not in Σ. Based on M we define a tree automaton $M'=(Q,\Sigma\cup\{d\},\delta',F)$ where δ' for elements of Σ is defined as δ and

$$\delta'(1,d)=11,\ \delta'(2,d)=22,\ \delta'(i,d)=1i \text{ for } 3\leq i\leq n.$$

The node labeled by d has two branches. If the root of the left branch is assigned with state 1 by M', then the node labeled by d is assigned with the same state (except state 2) as the root of its right branch. If the root of the left branch is assigned with state 2 by M', then the root of its right branch must also be assigned with state 2, and so is the node labeled by d. The sequence of states 12 is not defined at the node labeled by d. Now consider tree $t=d(t_1,t_2)$ where the roots of t_1 and t_2 are assigned with state $(\{1,2\},1)$ and $(\{1,2,\ldots,n\},2)$, respectively. According to the above definition, state $(\{0,1,\ldots,n-1\},n)$ is assigned to the root of the tree t. ■

Lemma 4. *All states in (1) are pairwise inequivalent.*

Proof. Let $(S_1,x)\neq(S_2,y)$ be two arbitrary states. First consider the case when neither of x and y is equal to n.

Assume $S_1\neq S_2$. Let $s\in S_1$ and $s\notin S_2$ (The other case is totally symmetric). According to the definition of the state in (1), $y\neq s$. After reading $n-s-1$ $a's$, a final state is reached from state (S_1,x), and no final state is reached from state (S_2,y).

Next we consider the case $S_1=S_2=S$, $S\neq\{0,1,\ldots,n-1\}$ and $x\neq y$. According to the definition of the state in (1), $x,y\in S$. Consider tree $T[X]=a^{n-1-x}(t,X)$ where state $(\{x,z\},z)$, $z\notin S$ is assigned to the root of subtree t. (Since $S\neq\{0,1,\ldots,n-1\}$, z always exists.) Let (S,x) and (S,y) be the states assigned to the roots of subtrees t_1 and t_2, respectively. Consider $T[X\leftarrow t_1]$. State $(\{x+1\},n)$ is assigned to the root of subtree $a(t,t_1)$. A final state $(\{n-1\},n)$ is assigned to the root of tree $a^{n-1-x}(t,t_1)$. Now consider $T[X\leftarrow t_2]$. State $(\emptyset,n)$ is assigned to the root of subtree $a(t,t_2)$. State $(\emptyset,n)$ is assigned to the root of tree $a^{n-1-x}(t,t_2)$, which is not a final state.

When $S_1=S_2=\{0,1,\ldots,n-1\}$ and $x\neq y$. After reading b, states (S_1,x) and (S_2,y) become to (S,x) and (S,y) where $S=\{0,2,3,\ldots,n-1\}$, respectively, which can be distinguished similarly as the above case.

Next consider the case when one of x and y is equal to n. Without loss of generality, assume $y = n$. Consider tree $T[X] = a^{n-1}(t, b^n(X))$ where state $(\{1,0\},1)$ is assigned to the root of subtree t. Let (S_1, x) and (S_2, n) be the states assigned to the roots of subtrees t_1 and t_2, respectively. Consider $T[X \leftarrow t_1]$. State $(\{0\},0)$ is assigned to the root of subtree $b^n(t_1)$. State $(\{1\},n)$ is assigned to the root of subtree $a(t, b^n(t_1))$. A final state $(\{n-1\},n)$ is assigned to the root of tree $a^{n-1}(t, b^n(t_1))$. Now consider $T[X \leftarrow t_2]$. State $(\{0\},n)$ is assigned to the root of subtree $b^n(t_2)$. State $(\emptyset,n)$ is assigned to the root of subtree $a(t, b^n(t_2))$. State $(\emptyset,n)$ is assigned to the root of tree $a^{n-1}(t, b^n(t_2))$, which is not a final state.

Now consider the case when $x = y = n$. Choose $s \in S_1$ and $s \notin S_2$ (The other case is totally symmetric). After reading $n-s-1$ $a's$, a final state is reached from state (S_1, n), and no final state is reached from state (S_2, n).

Thus, all states are pairwise inequivalent. ■

According to Lemma 3 and Lemma 4, we can get the following theorem.

Theorem 1. *Let $A = (Q, \Sigma, \delta, F)$ be an arbitrary DTA(DFA). We can construct a DTA(DFA) B recognizing language $L(A)^*_{(b)}$, $b \in \Sigma$ (bottom-up b-star operation) with at most $(|Q| + \frac{3}{2})2^{|Q|-1}$ vertical states.*

For any integer $n > 1$, there exists tree language L_n such that L_n can be recognized by a DTA(DFA) with at most n vertical states, and any DTA(DFA) recognizing $L^*_{n(b)}$ needs at least $(n + \frac{3}{2})2^{n-1}$ vertical states.

The lemma below gives the lower bound for top-down star operation.

Lemma 5. *The upper bound in lemma 2 is tight.*

Proof. The lemma can be similarly proved as Section 3 in [16], where we can view strings as unary trees. ■

4 Top-quotient and bottom-quotient

We have proved the following two theorems for top-quotient and bottom-quotient operations on unranked tree automata.

Theorem 2. *For any integer $n \geq 1$, n vertical states are necessary and sufficient in the worst case for a deterministic unranked tree automaton to accept the top-quotient of the tree language recognized by an n vertical states deterministic unranked tree automaton with respect to an arbitrary tree language T'.*

Proof. Let $A = (Q, \Sigma, \delta, F)$ where $|Q| = n$, be an arbitrary deterministic unranked tree automaton. We can construct a deterministic unranked tree automaton $B = (Q, \Sigma, \delta, E)$ recognizing $T' \top L(A)$, where $E = \{q \in Q \mid (\exists t' \in T')(\exists r \in (t' \cdot_b q))\ r^A \in F\}$.

Now we show that n states are necessary. Let $T' = \{b\}$ and $A = (\{q_1, q_2, \ldots, q_n\}, \{a\}, \delta, \{q_1\})$, where δ is defined as:

- for $1 \leq i \leq n-1$, $\delta(q_i, a) = q_{i+1}$,
- $\delta(q_n, a) = \epsilon$.

A accepts the unary tree language a^n. We have $T' \top L(A) = L(A)$. ■

Theorem 3. *For any integer $n \geq 1$, $(n+1)2^n - 1$ vertical states are necessary and sufficient in the worst case for a deterministic unranked tree automaton to accept the bottom-quotient of the tree language recognized by an n vertical states deterministic unranked tree automaton with respect to an arbitrary tree language T'.*

We prove Theorem 3 by stating two lemmas. Lemma 6 establishes the upper bound and Lemma 7 establishes a lower bound that differs by the constant one. Finally, after the proof of Lemma 7 we explain how the proof of Lemma 7 can be modified to exactly reach the upper bound given in Lemma 6.

Lemma 6. *For an arbitrary deterministic unranked tree automaton A with n vertical states and an arbitrary tree language T', any deterministic unranked tree automaton recognizing $L(A) \bot_b T'$ needs at most $(n+1)2^n - 1$ vertical states.*

Proof. Let $A = (Q, \Sigma, \delta, F)$ where $|Q| = n$, be an arbitrary deterministic unranked tree automaton. Define $G = \{t'^A \mid t' \in T'\}$. G consists of all the states that are assigned to the roots of trees in T' by A. We can construct a deterministic unranked tree automaton $B = (P, \Sigma, \mu, E)$ recognizing $L(A) \bot T'$, where

- $P = \{(q, \mathcal{P}(Q)) \mid q \in Q \cup \{dead\}\}$,
- $(q, S) \in E$ if $S \cap F \neq \emptyset$.

B operates as follows. For a leaf node u labeled by σ,

- assign $(q, \emptyset)$ to u if $\sigma \neq b$ and $\epsilon \in \delta(q, \sigma)$,
- assign (q, G) to u if $\sigma = b$ and $\epsilon \in \delta(q, b)$,
- assign $(dead, G)$ to u if $\sigma = b$ and $\epsilon \notin \delta(q, b)$.

For a tree $\sigma(\sigma_1, \ldots, \sigma_m)$ where the leaf nodes are assigned with a sequence of states $(q_1, S_1), \ldots, (q_m, S_m)$, the root node is assigned with state (q, S) provided that $q_1 \ldots q_m \in \delta(q, \sigma)$ and S consists of all $s \in Q$ such that for some $1 \leq i \leq m$

$$(\exists s_i \in S_i) \text{ such that } q_1 \ldots q_{i-1} s_i q_{i+1} \ldots q_m \in \delta(s, \sigma).$$

We define $q = \mathit{dead}$ if $q_1 \ldots q_m$ is not included in any $\delta(s, \sigma)$, $s \in Q$.

In states of B the first component directly simulates the computation of A. In a state (q, S), the second component keeps track of all states that A may be in provided that exactly one leaf labeled by b below the current node was replaced by a tree of T'.

Thus, the upper bound on the number of vertical states is $(n+1)2^n - 1$ as we do not require the tree automata to be complete. ■

Lemma 7. *There exists a tree language T that can be recognized by a deterministic unranked tree automaton with n vertical states, such that any deterministic unranked tree automaton recognizing $T \perp_b TREES_\Sigma$ needs at least $(n+1)2^n - 2$ vertical states.*

Proof.

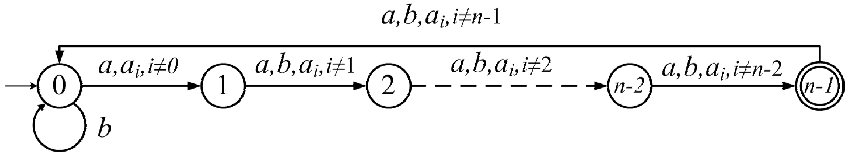

Fig. 3. Tree automaton A

Shown in Figure 3, A is the DFA we modified from the one used in Theorem 5 of [16] by adding symbols $a_0, \ldots, a_{n-1}$. Based on the DFA A we define the tree language T used in our lower bound construction. The tree language T consists of Σ-labeled trees t, $\Sigma = \{a, b, c\}$, where:

1. All leaves are labeled by a and if a node u has a child that is a leaf, then all the children of u are leaves.
2. A accepts the string of symbols labeling a path from any node of height one to the root.
3. The following holds for any $u \in \mathrm{dom}(t)$ and any nodes v_1 and v_2 of height one below u. If w_i is the string of symbols labeling the path from v_i to u, $i = 1, 2$, then A reaches the same state after reading strings w_1 and w_2.

The computation starts from nodes of height one. This is done because when the bottom-quotient takes place, the leaf node is substituted by a tree, and it will not appear in the resulting tree.

T can be recognized by a deterministic tree automaton $M = (Q, \{a, b, a_0, \ldots, a_{n-1}\}, \delta, F)$ where $Q = \{0, 1, \ldots, n-1\}$ and $F = \{n-1\}$. The transition function is defined as:

(1) $\delta(0, a) = \epsilon \cup (n-1)^+$,
(2) $\delta(i, a) = (i-1)^+$ for $1 \leq i \leq n-1$,
(3) $\delta(j, b) = (j-1)^+$ for $1 \leq i \leq n-1$,
(4) $\delta(0, b) = \{n-1, 0\}^+$,
(5) $\delta(i, a_j) = (i-1)^+$ for $1 \leq i \leq n-1$ and $i \neq j$,
(6) $\delta(0, a_j) = (n-1)^+$ for $j \neq n-1$.

Let $T' = TREES_\Sigma$ and N be the deterministic tree automaton for $L(M) \perp T'$ as considered in Lemma 6. According to the upper bound construction, the states in N are of the form $(q, S), 0 \leq q \leq n, S \subseteq \{0, 1, \ldots, n-1\}$, where $q = n$ denotes *dead* state in the construction.

First we show every state is reachable except $(n, \{0, 1, \ldots, n-1\})$. The leaf node labeled with a is assigned with $(0, \{0, 1, \ldots, n-1\})$. The state $(i, \{0, 1, \ldots, n-1\})$, $0 \leq i \leq n-1$ is reachable after reading a unary tree a^i. For $S \subseteq Q$, $0 \leq i \leq n-1$, $i \in S$, (i, S) is reachable from state (i, Q) by reading $s_0 s_{n-1} \ldots s_1$ where for each $0 \leq j \leq n-1$, $s_j = a$ if $j \in S$ and $s_j = b$, otherwise.

For $i \notin S$, we show that the state (i, S) is reachable using decreasing induction on the size of S. First we note that for $S \subseteq Q$, $0 \leq i \leq n-1$, $i \notin S$, $(i, Q - \{i\})$ is reachable from state $(i-1, Q)$ by reading a_i. Now suppose $(s_i, Q - \{s_1, \ldots, s_k\})$, $1 \leq i \leq k$ is reachable. Let $p = (s_j, Q - \{s_1, \ldots, s_{k+1}\})$, $1 \leq j \leq k+1$, $s_1 < \ldots < s_{k+1}$ and $Q - \{s_1, \ldots, s_{k+1}\} = \{r_1, \ldots, r_{n-k-1}\}$ be an arbitrary state. p is reachable from $(s_j - 1, \{r_1 - 1, r_2 - 1, \ldots, r_{n-k-1} - 1, s'\})$, by reading $a_{s'}$, where $s' \notin \{s_j - 1, r_1 - 1, r_2 - 1, \ldots, r_{n-k-1} - 1\}$.

We have already shown that $(0, \{0, 1, \ldots, n-1\})$ is reachable. State $(n, \{1, \ldots, n-1\})$ is reachable by reading a_0. State $(n, \{0, 1, \ldots, n-1\} - \{i\})$, $1 \leq i \leq n-1$ is reachable after reading i $a's$. Assume all states (n, S), $n > |S| \geq k+1$, $k < n-1$ are reachable. We will inductively show that all states where $|S| = k$ are reachable. Let (S, n) where $S = \{s_1, s_2, \ldots, s_k\}$, $0 \leq s_1 < s_2 < \ldots < s_k \leq n-1$ be an arbitrary state where $|S| = k$. According to the inductive assumption, state $(S \cup \{0\}, n)$ is reachable. State (S, n) is reachable by reading a_0. The state $(n, \emptyset)$ is reached at the root of tree $a(u, v)$ where u and v are assigned with $(1, \{1\})$ and $(2, \{2\})$, respectively.

That is all the states except $(n, \{0, 1, \ldots, n-1\})$ are reachable.

Now we show that all states are pairwise inequivalent. Given two states (i, S_1) and (j, S_2), first we consider the case when $S_1 \neq S_2$. Without loss of generality, let $s \in S_1 - S_2$. After reading unary tree a^{n-1-s}, a final state is reached from (i, S_1) and no final state is reached from (j, S_2). Now we consider the case when $i \neq j$ and $S_1 = S_2 = S$. Suppose (i, S) and (j, S) are assigned to the roots of tree t_1 and t_2, respectively. Consider tree $a^{n-2-i}(a(u, x), v)$ where u, v are assigned with states $(i, \{i\})$ and $(n, \{i+1\})$, respectively. Now $(i+1, \{i+1\})$ is assigned at the root of $a(u, x \leftarrow t_1)$ and $(n, \{i+2\})$ is assigned at the root of $a(a(u, x \leftarrow t_1), v)$. After reading the rest of a's, a final state $(n, \{n-1\})$ is reached. On the other hand, if $i \in S$, $(n, \{i+1\})$ is assigned at the root of $a(u, x \leftarrow t_2)$, $(n, \emptyset)$ if $i \notin S$. In either case $(n, \emptyset)$ is assigned at the root of $a(a(u, x \leftarrow t_1), v)$. $(n, \emptyset)$ is a dead state, which means no final state can be reached. ■

In fact, state $(n, \{0, 1, \ldots, n-1\})$ is reachable by introducing a new symbol and a few transitions. Let T be the tree language defined in Lemma 7 and M is the deterministic tree automaton constructed for T in the proof of Lemma 7. Let d be a new symbol not in Σ. Based on M we define a tree automaton $M' = (Q, \Sigma \cup \{d\}, \delta', F)$ where δ' for elements of Σ is defined as δ and

(1) $\delta(1, d) = 11$,
(2) $\delta(2, d) = 22$,
(3) $\delta(i, d) = 1i$ for $3 \leq i \leq n$.

The node labeled by d has two branches. If the root of the left branch is assigned with state 1 by M', then the node labeled by d is assigned with the same state (except state 2) as the root of its right branch. If the root of the left branch is assigned with state 2 by M', then the root of its right branch must also be assigned with state 2, and so is the node labeled by d. The sequence of states 12 is not defined at the node labeled by d. Now let $T' = TREES_{\Sigma \cup \{d\}}$. Consider tree $t = d(t_1, t_2)$ where the roots of t_1 and t_2 are assigned with state $(1, \{1, 2\})$ and $(2, \{1, 2, \ldots, n\})$, respectively. According to the above definition, state $(n, \{0, 1, \ldots, n-1\})$ is assigned to the root of the tree t.

We get a tight bound for the state complexity of top-quotient and bottom-quotient, respectively. Note that the bound for bottom-quotient differs, roughly, by a multiplicative factor $n+1$ from the corresponding result for ordinary finite automata. The precise state complexity of left-quotient for ordinary finite automata is $2^n - 1$ [16].

5 Conclusion

We have defined two star operations on tree languages according to the nature of tree structure, bottom-up and top-down star operations, and proved tight upper bounds on the number of vertical states for both star operations. The top-down star operation shares the state complexity of the star operation on string automata. On the other hand, we have established that the state complexity of the bottom-up star is $(n + \frac{3}{2})2^{n-1}$ which is of a different order than the corresponding result for the star operation on string automata. We defined top-quotient and bottom-quotient operations on trees and obtained tight state complexity bounds for both operations. The bound for the bottom-quotient operation differs by a multiplicative factor $n + 1$ from the corresponding result $2^n - 1$ for ordinary finite automata.

References

1. R. Alur, P. Madhusudan, Adding nesting structure to words, Lecture Notes in Computer Science, Volume 4036, pp. 1-13, 2006.
2. T. Bray, J. Paoli, C.M. Sperberg-McQueen, "Extensible Markup Language (XML) 1.0 (5th Edition)", W3C Recommendation, Nov. 2008, http://www.w3.org/TR/2008/REC-xml-20081126/.
3. J. Carme, J. Niehren, M. Tommasi, Querying unranked trees with stepwise tree automata, Lecture Notes in Computer Science, Volume 3091, pp. 105-118, 2004.
4. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, M. Tommasi, Tree automata techniques and applications, available on: http://www.grappa.univ-lille3.fr/tata, 1997, release October 1st, 2002.
5. H. Gruber, M. Holzer, Tight bounds for the descriptional complexity of regular expressions. *Proc. of DLT'09*, Lecture Notes in Computer Science, Volume 5583, pp. 276–287, 2009.
6. M. Holzer, M. Kutrib, Nondeterministic finite automata—Recent results on the descriptional and computational complexity. *Proc. of CIAA'08*, Lecture Notes in Computer Science, Volume 5148, pp. 1–16, 2008.
7. M. Holzer, M. Kutrib, Descriptional and computational complexity of finite automata, *Proc. of LATA'09*, Lecture Notes in Computer Science, Volume 5457, pp. 23–42, 2009.
8. D. Lee, M. Mani, M. Murata, Reasoning about XML schema languages using formal language theory, Technical Report, IBM Almaden Research Center, 2000.
9. M. Murata, D. Lee, M. Man, K. Kawaguchi, Taxonomy of XML schema languages using formal language theory, ACM (Association for Computing Machinery) Transactions on Internet Technology (TOIT), Volume 5, pp. 660-704, 2005.
10. X. Piao, K.Salomaa, Operational state complexity of deterministic unranked tree automata, *Proc. of DCFS'10,* 181-192.
11. X. Piao, K.Salomaa, Operational state complexity of nested word automata, Theoretical Computer Science, Volumn 410, pp. 3290-3302, 2009.

12. X. Piao, K.Salomaa, Transformations between different models of unranked bottom-up tree automata, *Proc. of DCFS'10,* 193-204.
13. T. Schwentick, Automata for XML-A survey, Journal of Computer and System Sciences, Volume 73, pp. 289-315, 2007.
14. S. Yu, Regular languages, in: Rozenberg, G., Salomaa, A. (eds.), *Handbook of Formal Languages,* volumn I, Springer, Berlin, pp. 41–110, 1997.
15. S. Yu, State complexity: Recent results and open problems, Fundamenta Informaticae, Volumn 64, pp. 471–481, 2005.
16. S. Yu, Q. Zhuang, K. Salomaa: The state complexity of some basic operations on regular languages, Theoretical Computer Science, Volumn 125, pp. 315-328, 1994.