Acta Cybernetica **22** (2015) 403–422.

State Complexity of Kleene-Star Operations on Regular Tree Languages[*]

Yo-Sub Han[†], Sang-Ki Ko[†], Xiaoxue Piao[‡], and Kai Salomaa[‡]

Dedicated to the memory of Professor Ferenc Gécseg (1939–2014)

Abstract

The concatenation of trees can be defined either as a sequential or a parallel operation, and the corresponding iterated operation gives an extension of Kleene-star to tree languages. Since the sequential tree concatenation is not associative, we get two essentially different iterated sequential concatenation operations that we call the bottom-up star and top-down star operation, respectively. We establish that the worst-case state complexity of bottom-up star is $(n + \frac{3}{2}) \cdot 2^{n-1}$. The bound differs by an order of magnitude from the corresponding result for string languages. The state complexity of top-down star is similar as in the string case. We consider also the state complexity of the star of the concatenation of a regular tree language with the set of all trees.

Keywords: tree automata, state complexity, iterated concatenation

1 Introduction

The descriptional complexity of finite automata has been studied for over half a century [13, 15, 16], and there has been particularly much work done over the last two decades. The reader may find more information in the surveys [4, 8, 12]. Also the state complexity of various extensions of finite automata, such as tree automata [14, 19] and input-driven pushdown automata (a.k.a. nested word automata) [7, 17] has been considered. These models retain the feature of finite

[*]A preliminary version of parts of this paper appeared in the proceedings of *Computation, Physics and Beyond, International Workshop on Theoretical Computer Science, WTSC2012.* Han and Ko were supported by the Basic Science Research Program through NRF funded by MEST (2012R1A1A2044562) and the International Cooperation Program managed by NRF of Korea (2014K2A1A2048512). Piao and Salomaa were supported by the Natural Sciences and Engineering Research Council of Canada Grant OGP0147224.

[†]Department of Computer Science, Yonsei University, Seoul 120-749, Republic of Korea, E-mail: {emmous, narame7}@cs.yonsei.ac.kr

[‡]School of Computing, Queen's University, Kingston, Ontario K7L 2N8, Canada, E-mail: {piao,ksalomaa}@cs.queensu.ca

DOI: 10.14232/actacyb.22.2.2015.11

automata that a nondeterministic automaton can be converted to an equivalent deterministic automaton.

Concatenation of tree languages can be defined either as a sequential or a parallel operation. Tight state complexity bounds for the concatenation of regular (respectively, subtree-free) tree languages were given in [18] (respectively, [3]) and the state complexity of concatenation operations with the set of all trees was considered in [11].

Here we consider the iterated concatenation of trees, that is, an extension of the Kleene-star operation for tree languages. If defined in the usual way, the iterated parallel concatenation is not a regularity preserving operation and Gécseg and Steinby [6] define the Kleene-star of tree languages slightly differently. Since sequential concatenation of tree languages is non-associative, there are two essentially different ways to define the corresponding iterated operation. We name these variants the *bottom-up star* and the *top-down star* operations. It is easy to see that the top-down (sequential) star operation coincides with the iterated product (Kleene-star) based on parallel concatenation considered in [6].

We give tight state complexity bounds for both the bottom-up and the top-down Kleene-star operations. We show that the bottom-up star of a tree language recognized by a deterministic bottom-up automaton with n states can be recognized by an automaton with $(n + \frac{3}{2}) \cdot 2^{n-1}$ states and, furthermore, there exist worst-case examples where this number of states is needed. This bound is, roughly, n times the corresponding bound for regular string languages. On the other hand, the state complexity of the top-down star operation is shown to coincide with the state complexity of Kleene-star on string languages.

The state complexity of combined operations on regular languages was first considered by A. Salomaa et al. [21], and later there has been much interest in this topic [2, 10]. In the last section we consider the state complexity of tree concatenation combined with star in the special case where one of the argument languages consists of the set of all trees. For some of the combined operations we get tight bounds that are significantly lower than the function composition of the state complexity of concatenation with F_Σ and the state complexity of the corresponding star operation.

To conclude the introduction we comment on the difference between classical ranked tree automata [5] and unranked tree automata. Much of the recent work on tree automata uses automata operating on unranked trees that are used in modern applications such as XML document processing [1, 18, 19, 22]. The transitions of an unranked tree automaton A are defined in terms of regular languages, called *horizontal languages.* Each horizontal language is specified by a deterministic finite automaton (DFA) that processes strings of states of the bottom-up computation, or *vertical states.* The size of A is defined to be the sum of the number of vertical states and the numbers of states of the DFAs used to define the horizontal languages.

In the case of the Kleene-star operations, the worst-case state complexity bounds for the numbers of vertical states can be reached using just binary trees, and for the sake of readability we restrict here consideration to automata operating on ranked trees. The upper bound construction for bottom-up star for unranked tree automata was given in [20]. The generalized construction relies on the same ideas as Lemma 2 below, however, the notations are considerably more involved.

In the case of DFAs operating on strings, it is common to give state complexity bounds in terms of complete DFAs, that is, all transitions of a DFA are required to be defined, see e.g. [8, 24]. In order to keep our state complexity bounds consistent with corresponding results for tree automata operating on unranked trees [1, 18, 19], our definition allows a deterministic tree automaton to have undefined transitions.

Note that requiring a ranked tree automaton (or an ordinary DFA) to be complete, changes the number of states by at most one. On the other hand, for deterministic tree automata operating on unranked trees where the horizontal languages are defined by DFAs [1, 18, 19], the sizes of an incomplete deterministic automaton and the corresponding completed version may be significantly different. In an unranked tree automaton, adding a dead state q_{sink} for the bottom-up computation, requires the addition, corresponding to an input symbol σ, a horizontal language $L_{\sigma,q_{\text{sink}}}$ that is the complement of a finite disjoint union $L_{\sigma,q_1} \cup \ldots \cup L_{\sigma,q_n}$, where $q_1, \ldots, q_n$ are the vertical states of the incomplete automaton. The size of the minimal DFA for $L_{\sigma,q_{\text{sink}}}$ may be considerably larger than the sum of the sizes of the DFAs for L_{σ,q_i}, $i = 1, \ldots, n$, [9].

2 Basic definitions on tree automata

We assume that the reader is familiar with the basics of automata and formal languages [23, 24]. Here we recall and introduce some definitions related to tree automata. For more information the reader may consult the texts by Gécseg and Steinby [5, 6] or the electronic book by Comon et al. [1].

The cardinality of a finite set S is $|S|$ and the power set of S is 2^S. The set of positive integers is $\mathbb{N}$. A ranked alphabet is a finite set Σ where each element is associated a nonnegative integer as its rank. The set of elements of rank m is Σ_m, $m \geq 0$. The set of trees over ranked alphhbet Σ, or Σ-trees, F_Σ, is the smallest set S satisfying the condition: if $m \geq 0$, $\sigma \in \Sigma_m$ and $t_1, \ldots, t_m \in S$ then $\sigma(t_1, \ldots, t_m) \in S$.

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 $1 \leq j < i$. The set of nodes of a tree $t \in F_\Sigma$ can be represented in the well-known way as a tree domain $\text{dom}(t) \subseteq \{1, \ldots, M\}^*$ where M is the largest rank of any element of the ranked alphabet Σ. The tree t is then viewed as a mapping $t : \text{dom}(t) \to \Sigma$.

We assume that notions such as the *root,* a *leaf,* a *subtree* and the *height* of a tree are known. We use the convention that the height of a single node tree is zero. For $\sigma \in \Sigma$ and $t \in F_\Sigma$, $\text{leaf}(t, \sigma) \subseteq \text{dom}(t)$ denotes the set of leaves of t with label σ. Let t be a tree and u some node of t. The tree obtained from t by replacing the subtree at node u with a tree s is denoted $t(u \leftarrow s)$. The notation is extended in the natural way for a set of pairwise independent nodes U of t and $S \subseteq F_\Sigma$: $t(U \leftarrow S)$ is the set of trees obtained from t by replacing each node of U by some tree in S.

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

A *deterministic bottom-up tree automaton* (DTA) is a tuple $A = (\Sigma, Q, Q_F, g)$, where Σ is a ranked alphabet, Q is a finite set of states, $Q_F \subseteq Q$ is a set of accepting states and g associates to each $\sigma \in \Sigma_m$ a partial function $\sigma_g : Q^m \longrightarrow Q$, $m \geq 0$. In the usual way, we define the state $t_g \in Q$ reached by A at the root of a tree $t = \sigma(t_1, \ldots, t_m)$, $\sigma \in \Sigma_m$, $m \geq 0$, $t_i \in F_\Sigma$, $i = 1, \ldots, m$, inductively by setting $t_g = \sigma_g((t_1)_g, \ldots, (t_m)_g)$ if the right side is defined, and t_g is undefined otherwise. The tree language recognized by A is $L(A) = \{t \in F_\Sigma \mid t_g \in Q_F\}$. Deterministic bottom-up tree automata recognize the family of regular tree languages.

The intermediate stages of a computation of A, called *configurations of A,* are Σ-trees where some leaves may be labeled by states of A. The set of configurations of A consists of Σ^A-trees where $\Sigma_0^A = \Sigma_0 \cup \{Q\}$ and $\Sigma_m^A = \Sigma_m$ when $m \geq 1$.

A bottom-up automaton begins processing the tree from the leaves because, following a common custom, we view trees to be drawn with the root at the top. As discussed in the previous section, our definition allows a DTA to have undefined transitions, that is, σ_g, $\sigma \in \Sigma_m$, is a partial function.

2.1 Iterated concatenation of trees

We extend the string concatenation operation to an operation where a leaf of a tree is replaced by another tree. Concatenation of trees can be defined also as a parallel operation, however, as will be observed below the iteration of parallel concatenation does not preserve recognizability.

For $\sigma \in \Sigma_0$ and $t_1, t_2 \in F_\Sigma$, we define the *sequential σ-concatenation* of t_1 and t_2 as

$$t_1 \cdot_\sigma^s t_2 = \{\ t_2(u \leftarrow t_1) \mid u \in \text{leaf}(t_2, \sigma)\ \}. \tag{1}$$

That is, $t_1 \cdot_\sigma^s t_2$ is the set of trees obtained from t_2 by replacing one occurrence of a leaf labeled by σ with t_1. The definition is extended in the natural way for tree languages $T_1, T_2 \subseteq F_\Sigma$ by setting

$$T_1 \cdot_\sigma^s T_2 = \bigcup_{t_i \in T_i, i=1,2} t_1 \cdot_\sigma^s t_2.$$

Alternatively, we can consider a *parallel σ-concatenation* of tree languages $T_1, T_2 \subseteq F_\Sigma$ by setting

$$T_1 \cdot_\sigma^p T_2 = \{\ t_2(\text{leaf}(t_2, \sigma) \leftarrow T_1) \mid t_2 \in T_2\ \}.$$

The operation $T_1 \cdot_\sigma^p T_2$ is called the σ-product of T_1 and T_2 in [6]. Note that the parallel concatenation of tree languages could not be defined by defining first the concatenation of individual trees (as was done for sequential concatenation in (1)) and then taking union over sets of trees. For trees $t_1, t_2 \in F_\Sigma$, $t_1 \cdot_\sigma^p t_2$ is an individual tree while $t_1 \cdot_\sigma^s t_2$ is a set of trees. In the case where no leaf of t_2 is labeled by σ, $t_1 \cdot_\sigma^s t_2 = \emptyset$ and $t_1 \cdot_\sigma^p t_2 = t_2$.

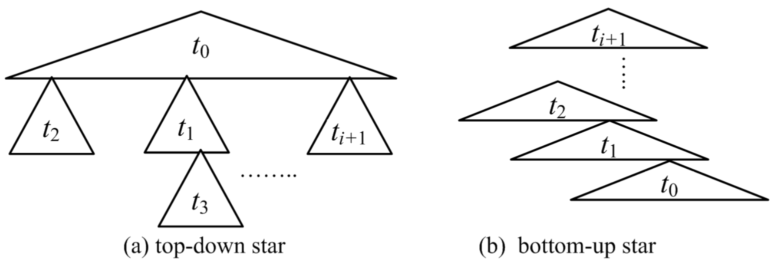

Figure 1: A tree in $T_\sigma^{s,t,*}$ (a) and in $T_\sigma^{s,b,*}$ (b). Here $t_0, t_1, \ldots t_{i+1}$ are trees in T.

When considering bottom-up tree automata operating on unary trees, both of the above definitions reduce to the usual concatenation of string languages: when processing $T_1 \circ T_2$, $\circ \in \{\cdot_\sigma^s, \cdot_\sigma^p\}$, the automaton reads first an element of T_1 and then an element of T_2.

The parallel concatenation operation is associative, however, sequential concatenation is nonassociative, as observed below in Example 1. The nonassociativity of sequential concatenation means, in particular, that there are two variants of the iteration of the operation.

For $\sigma \in \Sigma$ and $T \subseteq F_\Sigma$, we define the *kth sequential top-down σ-power of T,* $k \geq 0$, by setting $T_\sigma^{s,t,0} = \{\sigma\}$, and $T_\sigma^{s,t,k} = T \cdot_\sigma^s T_\sigma^{s,t,k-1}$, when $k \geq 1$. The *sequential top-down σ-star of T* is then

$$T_\sigma^{s,t,*} = \bigcup_{k \geq 0} T_\sigma^{s,t,k}.$$

Similarly, the *kth sequential bottom-up σ-power of T,* is defined by setting $T_\sigma^{s,b,0} = \{\sigma\}$, $T_\sigma^{s,b,1} = T$ and $T_\sigma^{s,b,k} = T_\sigma^{s,b,k-1} \cdot_\sigma^s T$, when $k \geq 2$. The *sequential bottom-up σ-star of T* is

$$T_\sigma^{s,b,*} = \bigcup_{k \geq 0} T_\sigma^{s,b,k}.$$

Note that the definition of bottom-up σ-powers explicitly sets $T_\sigma^{s,b,1}$ to be equal to T. This is done because $T_\sigma^{s,b,0} \cdot_\sigma^s T$ can be a strict subset of T if some trees of T contain no occurrences of σ. Figure 1 illustrates the definitions of top-down star and bottom-up star.

Example 1. It is easy to see that sequential concatenation is non-associative. Consider a ranked alphabet Σ determined by $\Sigma_2 = \{\omega\}$, $\Sigma_0 = \{\sigma\}$ and let $t = \omega(\sigma, \sigma)$. Now $t \cdot_\sigma^s t = \{\omega(\omega(\sigma,\sigma),\sigma), \omega(\sigma, \omega(\sigma,\sigma))\}$ and $t_1 = \omega(\omega(\sigma,\sigma), \omega(\sigma,\sigma)) \in t \cdot_\sigma^s (t \cdot_\sigma^s t)$ but, on the other hand, $t_1 \notin (t \cdot_\sigma^s t) \cdot_\sigma^s t$.

To illustrate the difference of top-down and bottom-up star, respectively, consider $T = \{\omega(\sigma, \sigma)\}$. We note that $T_\sigma^{s,t,*} = F_\Sigma$ and

$$T_\sigma^{s,b,*} = \{r \in F_\Sigma \mid \text{ each non-leaf node of } r \text{ has at least one leaf as a child } \}.$$

Note that with $T = \{\omega(\sigma, \sigma)\}$, $T_\sigma^{s,b,k}$, $k \geq 0$, consists of trees of height (exactly) k. The trees of $T_\sigma^{s,b,*}$ all consist of a path labeled by binary symbols ω and all children of nodes of the path that "diverge" from the path are labeled by the leaf symbol σ.

The following characterization of bottom-up σ-star as the smallest set closed under concatenation with T from the right follows directly from the definition of bottom-up star. The characterization will be used in the next section.

Lemma 1. *For $\sigma \in \Sigma_0$ and $T \subseteq F_\Sigma$, define $\mathrm{cl}_\sigma(T)$ as the smallest set $S \subseteq F_\Sigma$ such that (i) $T \cup \{\sigma\} \subseteq S$, and (ii) $t_1 \cdot_\sigma^s t_2 \in S$ for every $t_2 \in T$ and $t_1 \in S$. Then $\mathrm{cl}_\sigma(T) = T^{s,b,*}$.*

Completely analogously we can define, for $T \subseteq F_\Sigma$, the parallel σ-star of T, denoted $T_\sigma^{p,*}$. Since parallel concatenation is associative, we do not need to distinguish the bottom-up and top-down variants. However, we note that with $T = \{\omega(\sigma, \sigma)\}$, $T_\sigma^{p,*}$ consists of all balanced trees over the ranked alphabet Σ, where $\Sigma_2 = \{\omega\}$, $\Sigma_0 = \{\sigma\}$. Since the "straightforward" definition of Kleene-star based on parallel concatenation does not preserve regularity, in fact, Gécseg and Steinby [6] define a regularity preserving σ-iteration operation by defining the kth ($k \geq 1$) power of T by parallel-concatenating the union of all the ith powers of T, $0 \leq i \leq k-1$, with the tree language T.

It is easy to verify that the definition of the σ-iteration operation (based on parallel concatenation) given in section 7 of [6] coincides with the sequential top-down star defined above, and in the following we will focus only on the sequential variants of iterated concatenation. The top-down (respectively, bottom-up) σ-powers and σ-star of a tree language T are in the following denoted $T_\sigma^{t,k}$, ($k \geq 0$), and $T_\sigma^{t,*}$ (respectively, $T_\sigma^{b,k}$ and $T_\sigma^{b,*}$), that is, we drop the superscript "s" in the notation.

3 Bottom-up and top-down star: state complexity

We establish for the bottom-up star operation a tight state complexity bound that is of a different order of magnitude than the state complexity of Kleene-star for string languages. First we give an upper bound for the state complexity of bottom-up star.

Lemma 2. *Suppose that tree language L is recognized by a DTA with n states. For $\sigma \in \Sigma_0$, the tree language $L_\sigma^{b,*}$ can be recognized by a DTA with $(n + \frac{3}{2})2^{n-1}$ states.*

Proof. Let $A = (\Sigma, Q, Q_F, g_A)$ be a DTA with n states recognizing the tree language L. Without loss of generality we assume that σ_{g_A} is defined, because otherwise

$$L(A)_\sigma^{b,*} = L(A)_\sigma^{b,0} \cup L(A)_\sigma^{b,1} = \{\sigma\} \cup L(A),$$

and it is easy to construct a DTA with $n+1$ states that recognizes $L(A) \cup \{\sigma\}$.

Choose three disjoint subsets of $2^Q \times (Q \cup \{\text{dead}\})$ by setting

(i) $P_1 = \{(S, q) \mid S \in 2^Q, \{q, \sigma_{g_A}\} \subseteq S, q \in Q_F\}$,

(ii) $P_2 = \{(S, q) \mid S \in 2^Q, q \in S \cap (Q - Q_F)\}$,

(iii) $P_3 = \{(S, \text{dead}) \mid S \in 2^Q, S \neq \emptyset\}$.

Here dead is a new element not in Q. Now define a DTA $B = (\Sigma, P, P_F, g_B)$ where

$$P = P_1 \cup P_2 \cup P_3 \cup \{p_{\text{new}}\}, \quad P_F = \{(S, q) \in P \mid S \cap Q_F \neq \emptyset\} \cup \{p_{\text{new}}\}.$$

We define the transitions of B by setting, $\sigma_{g_B} = p_{\text{new}}$, and for $\tau \in \Sigma_0 - \{\sigma\}$,

$$\tau_{g_B} = \begin{cases} (\{\tau_{g_A}, \sigma_{g_A}\}, \tau_{g_A}) & \text{if } \tau_{g_A} \in Q_F, \\ (\{\tau_{g_A}\}, \tau_{g_A}) & \text{if } \tau_{g_A} \in Q - Q_F, \\ \text{undefined}, & \text{if } \tau_{g_A} \text{ is undefined.} \end{cases} \tag{2}$$

To define transitions on Σ_m, $m \geq 1$, we view p_{new} as the state $(\{\sigma_{g_A}\}, \sigma_{g_A})$, and hence every state of B is represented in the form (S, q), $S \subseteq Q$, $q \in Q$. (Note that p_{new} is not the same as $(\{\sigma_{g_A}\}, \sigma_{g_A})$, because the former is an accepting state and the latter need not be accepting.) For $\tau \in \Sigma_m$ and $(S_1, q_1), \ldots, (S_m, q_m) \in P$, we first denote

$$X = \bigcup_{i=1}^{m} \{\tau_{g_A}(q_1, \ldots, q_{i-1}, z, q_{i+1}, \ldots, q_m) \mid z \in S_i\}$$

Now we define

$$\tau_{g_B}((S_1, q_1), \ldots, (S_m, q_m)) \tag{3}$$

to be equal to

(i) $(X \cup \{\sigma_{g_A}\}, \tau_{g_A}(q_1, \ldots, q_m))$ if $\tau_{g_A}(q_1, \ldots, q_m) \in Q_F$,

(ii) $(X, \tau_{g_A}(q_1, \ldots, q_m))$ if $\tau_{g_A}(q_1, \ldots, q_m) \in Q - Q_F$,

(iii) (X, dead) if $X \neq \emptyset$ and $\tau_{g_A}(q_1, \ldots, q_m)$ is undefined.

In the remaining case, where $X = \emptyset$ and $\tau_{g_A}(q_1, \ldots, q_m)$ is undefined, also (3) is undefined. Note that if for some $1 \leq i \leq m$, $q_i = \text{dead}$, this implies automatically that $\tau_{g_A}(q_1, \ldots, q_m)$ is undefined.

Recall that if (S, q), $S \subseteq Q$, $q \in Q$ is a state of B then $q \in S$ and, furthermore, if $q \in Q_F$ then $\sigma_{g_A} \in S$. The transitions of g_B preserve this property and the state in (i) (in (ii), (iii), respectively) is an element of P_1 (an element of P_2, P_3, respectively).

The second component of the state of B simply simulates the computation of A on the current subtree, and goes to the state dead if the next state of A is undefined. Intuitively, the first component of the state of B consists of all states that A could reach at the current subtree t' assuming that

$$\text{in } t' \text{ at most one subtree of } L(A)_{\sigma}^{b,k},\ k \geq 0, \text{ has been replaced by a leaf } \sigma. \tag{4}$$

Inductively, assume that B assigns to the root of tree t_i a state $(S_i, (t_i)_{g_A})$ where $S_i \subseteq Q$ satisfies the property (4) for t_i, $i = 1, \ldots, m$. Now the rule (3) assigns to the

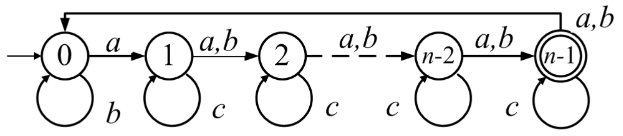

Figure 2: The DFA A from [24] with added c-transitions.

root of tree $t = \tau(t_1, \ldots, t_m)$ a state (S, q) where $q = \tau_{g_A}((t_1)_{g_A}, \ldots, (t_m)_{g_A})$ and S consists of all states that A could reach at the root of t assuming the computation uses as arguments $q_1, \ldots, q_m$ where (by the definition of the set X) at most one of the q_i's can be replaced by an arbitrary state from S_i, $1 \leq i \leq m$. This means that the state (S, q) again satisfies the property (4) for the tree t.

The choice of the set of final states P_F and Lemma 1 now imply that $L(B) = L(A)^{b,*}_{\sigma}$.

It remains to estimate the worst-case size of B. We note that if $Q_F = \{\sigma_{g_A}\}$, in B only states of the form $(\{q\}, q)$, $q \in Q$, can be reachable, and p_{new} can be identified with $(\{\sigma_{g_A}\}, \sigma_{g_A})$. In this case $L(A)^{b,*}_{\sigma}$ has a DTA with n states. Thus, without loss of generality we assume that Q_F contains a final state distinct from σ_{g_A}.

We note that $|P_1| = |Q_F| \cdot 2^{n-2}$, $|P_2| = |Q - Q_F| \cdot 2^{n-1}$ and $|P_3| = 2^n - 1$. Here the estimation of the size of P_1 relies on the above observation that we can exclude the possibility $Q_F = \{\sigma_{g_A}\}$. Thus, the cardinality of $P_1 \cup P_2 \cup P_3 \cup \{p_{\text{new}}\}$ is maximized as $(n + \frac{3}{2})2^{n-1}$ when $|Q_F| = 1$. $\square$

The upper bound of Lemma 2 is of a different order of magnitude than the known state complexity of Kleene-star for string languages [24]. It remains to verify that the bound of Lemma 2 can be reached in the worst case.

Figure 2 represents a DFA A used in [24, 25] for the lower bound construction for Kleene-star where we have added transitions on the symbol c. Note that A is an incomplete DFA since the c transition on 0 is undefined. Based on A we define in the following a tree automaton M_A.

Choose $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$ where $\Sigma_0 = \{e\}$, $\Sigma_1 = \{a, b, c\}$ and $\Sigma_2 = \{a_2, d_2\}$. We define a DTA $M_A = (\Sigma, Q_A, Q_{A,F}, g_A)$, where $Q_A = \{0, 1, \ldots, n-1\}$, $Q_{A,F} = \{n-1\}$ and the transition function g_A is defined by setting:

(i) $e_{g_A} = 0$, $c_{g_A}(i) = i$, $1 \leq i \leq n-1$,

(ii) $a_{g_A}(i) = (a_2)_{g_A}(i, i) = i + 1$, $0 \leq i \leq n-2$,
$a_{g_A}(n-1) = (a_2)_{g_A}(n-1, n-1) = 0$,

(iii) $b_{g_A}(i) = i + 1$, $1 \leq i \leq n-2$, $b_{g_A}(j) = 0$, $j \in \{0, n-1\}$,

(iv) $(d_2)_{g_A}(0, i) = i$, $i = 0, 2, 3, \ldots, n-1$, $(d_2)_{g_A}(1, 1) = 1$.

All transitions of g_A not listed above are undefined. Intuitively, the construction of M_A can be, roughly speaking, explained as follows. Denote by T_d the subset of

F_Σ consisting of trees without any occurrences of the binary symbol d_2, thus the only binary symbol in trees of T_d is a_2. On a tree $t \in T_d$, the DTA M_A simulates the computation of A on each string of symbols starting from a node of height one, where occurrences of a_2 are "interpreted" simply as a. The computations on different paths verify that for any $u \in \mathrm{dom}(t)$ labeled by a_2 and any nodes v_1 and v_2 of height one below u, the simulated computations started from v_1 and v_2 agree at u.

Note that the original DFA has no transitions on d, and the transitions on d_2 have been added for a technical reason that will be used in the proof of Lemma 4. Also, the above intuitive description is not completely precise on how M_A operates on binary symbols a_2 where one child is a leaf (that gets assigned the state 0) and the other child is not a leaf. The following Lemmas 3 and 4 rely only on the formal definition of the transition function g_A of M_A. The above intuitive description of the operation of M_A is intended only as a guide that may be useful in understanding the operation of the DTA constructed to recognize the bottom-up e-star of $L(M_A)$. Finally, note that the d_2-transitions will be needed only to establish the reachability of one particular state, and in most of the technical constructions the above intuitive description of the operation of M_A (based on the DFA A of Figure 2) can be sufficient.

Using the construction of the proof of Lemma 2, based on M_A we construct a DTA $M_B = (\Sigma, Q_B, Q_{B,F}, g_B)$ that recognizes the tree language $L(M_A)_e^{b,*}$. We make the convention that the sink-state "dead" used in the proof is denoted by n. Thus the set of states Q_B consists of the special state p_{new} assigned to e and all pairs

$$(P, q),\ P \subseteq \{0, \ldots, n-1\},\ 0 \le q \le n, \tag{5}$$

where $0 \le q \le n-1$ implies $q \in P$, $q = n-1$ implies $0 \in P$ and $q = n$ implies $P \neq \emptyset$. The number of pairs as in (5) is $(n + \frac{3}{2})2^{n-1} - 1$.

In the following two lemmas we establish that M_B is a minimal DTA. That is, first we show that all states of Q_B are pairwise inequivalent with respect to the Myhill-Nerode equivalence relation extended to trees. Second we show that all states of Q_B are reachable, that is, for each $q \in Q_B$ there exists $t \in F_\Sigma$ such that $t_{g_B} = q$. The proof of our first lemma assumes that all states are reachable which will be established next in Lemma 4[1].

Lemma 3. *All states of M_B are pairwise inequivalent.*

Proof. For the sake of convenience, we assume that we have already proven that all states of M_B are reachable (Lemma 4). Thus, in order to distinguish two states with respect to the Myhill-Nerode relation, we can use an arbitrary configuration of M_B where one leaf is replaced by the given states. More formally, in order to show that two distinct states of Q_B, p_1 and p_2, are inequivalent, it is sufficient to find $t \in F_{\Sigma^{M_B}}[x]$ such that the computation of M_B started from the configuration $t(x \leftarrow p_1)$ accepts if and only if the computation started from the configuration $t(x \leftarrow p_2)$ does not accept.

[1]The proof of Lemma 4 does not rely on Lemma 3.

We first show that any two distinct states (S_1, q_1) and (S_2, q_2) as in (5) are not equivalent. After that we consider the special state p_{new}. We begin by considering the case where neither of q_1 or q_2 is equal to n (which was used to denote the dead state of M_A).

Case $0 \leq q_1, q_2 \leq n-1$: (a) Assume $S_1 \neq S_2$ and $s \in S_1 - S_2$ (The other possibility is completely symmetric.) After reading $n-s-1$ unary symbols a, a final state is reached from state (S_1, q_1). On the other hand, since (S_2, q_2) is as in (5), $q_2 \neq s$. This means that the computation C that begins with (S_2, q_2) and reads $n-s-1$ unary symbols a ends with a non-final state. Note that at some point during the computation C, the second component may become $n-1$ which adds an element 0 to the first component. However, at the end of the computation C the first component cannot contain $n-1$.

(b)(i) Next we consider the case $S_1 = S_2 = S$, $\{0, 1, \ldots, n-2\} \not\subseteq S$ and $q_1 \neq q_2$. According to the definition of the states (5), $q_1, q_2 \in S$. Choose $p \in \{0, 1, \ldots, n-2\} - S$ and consider a tree $t_1 = a^{2n-2-q_1} a_2((\{q_1, p\}, p), x) \in F_{\Sigma^{M_B}}[x]$. Since $p \in \{0, 1, \ldots, n-2\}$, $(\{q_1, p\}, p)$ is a legal state (5). Consider the computation of M_B on tree $t_1(x \leftarrow (S, q_1))$. Since $p \notin S$ the state $(\{q_1+1\}, n)$ is assigned to the root of the subtree $a_2((\{q_1, p\}, q_1), (S, q_1))$. (Here addition is modulo n.) After this the computation reads the $2n-2-q_1$ unary symbols a in t_1 and ends in an accepting state. On the other hand, consider the computation of M_B on $t_1(x \leftarrow (S, q_2))$. Since $p \notin S$ and $q_2 \notin \{q_1, p\}$, the transition $(a_2)_{g_B}$ on arguments $(\{q_1, p\}, p)$, $(S, q_2))$ is undefined and the computation does not accept.

(b)(ii) Consider $S = \{0, 1, \ldots, n-2\}$, and hence we know that $q_1, q_2 \neq n-1$. From state (S, q_i) by reading a unary symbol b we get (S', q_i'), where $S' = \{0, 2, \ldots, n-2, n-1\}$. Since $q_1, q_2 \neq n-1$, $q_1' \neq q_2'$ and the states (S', q_1') and (S', q_2') are distinguished as in b(i) above.

(b)(iii) Consider then the possibility $S = \{0, 1, \ldots, n-1\}$ and $q_1 \neq q_2$. If $\{q_1, q_2\} \neq \{0, n-1\}$, by reading a unary symbol b from (S, q_1) and (S, q_2), respectively, we get two states (S', q_1'), (S', q_2'), $q_1' \neq q_2'$, that are distinguished as in the previous case[2]. Next consider the case $\{q_1, q_2\} = \{0, n-1\}$, and first assume that $n \geq 3$. By reading a unary symbol a we obtain states (S, q_1+1), (S, q_2+1) where $q_1+1 \neq q_2+1$ and $q_i+1 \neq n-1$, $i = 1, 2$ (addition is modulo n). The states (S, q_1+1) and (S, q_2+1) can be distinguished as in the previous cases.

Finally consider the possibility $n = 2$ and $\{q_1, q_2\} = \{0, 1\}$. From state $(\{0, 1\}, 1)$ by reading unary symbols ca, we reach the accepting state $(\{0, 1\}, 0)$. On the other hand, a computation starting from $(\{0, 1\}, 0)$ by reading the unary symbols ca reaches the nonaccepting state $(\{0\}, 2)$.

Case where $q_2 = n$: First assume $q_1 \neq n$. Choose $t_2 \in F_{\Sigma^{M_B}}[x]$ by setting $t_2 = a^{n-2} a_2((\{0, 1\}, 1), b^{n-1}(x))$. Since $n-1$ consecutive b-transitions take any

[2]The b-transitions of A violate injectivity only on states 0 and $n-1$.

state of A to state 0, the computation of M_B on $t_2(x \leftarrow (S_1, q_1))$ assigns state $(\{0\}, 0)$ to the root of the subtree $b^{n-1}((S_1, q_1))$. Then the state $(\{1\}, n)$ is reached at the root of the subtree $a_2((\{0, 1\}, 1), b^{n-1}((S_1, q_1)))$. A final state $(\{n-1\}, n)$ is reached after reading further $n-2$ unary symbols a. On the other hand, in the computation of M_B on $t_2(x \leftarrow (S_2, n))$ the state $(\{0\}, n)$ is assigned to the root of the subtree $b^{n-1}((S_2, n))$. When reading the binary symbol a_2 with arguments $(\{0, 1\}, 1)$ and $(\{0\}, n)$ the computation step of M_B is undefined, and hence M_B does not accept $t_2(x \leftarrow (S_2, n))$.

Finally consider the case where also $q_1 = n$. Thus $S_1 \neq S_2$ and choose $s \in S_1 - S_2$. After reading $n - s - 1$ unary symbols a, a final state is reached from state (S_1, n), and the same computation does not reach a final state from (S_2, n).

It remains to show that p_{new} is not equivalent with any state (S, q) as in (5). Since p_{new} is final, it is sufficient to consider states where $n - 1 \in S$. Thus, by reading a unary symbol c from state (S, q) we get a state (S', q'), where $n - 1 \in S'$ and $0 \leq q' \leq n$. On the other hand, computations starting from p_{new} are identical to computations starting from $(\{0\}, 0)$ and hence a computation step with unary symbol c is undefined. □

Before the next lemma we introduce the following notation. For a unary tree representing a configuration of M_B, $t = z_1(z_2(\ldots z_m(z_0)\ldots)) \in F_{\Sigma^{M_B}}$, we define $\text{word}(t) = z_m z_{m-1} \ldots z_1$. Note that $\text{word}(t)$ consists of the sequence of symbols labeling the nodes of t bottom-up, and the label of the leaf is not included. In the following when we refer to $\text{word}(t)$ of a tree t, without further mention, this implies that t is a unary tree.

Lemma 4. *All states of M_B are reachable.*

Proof. The transition function of M_B assigns the special state p_{new} to leaf symbol e. Recall that from p_{new} the computation of M_B continues as from $(\{0\}, 0)$. Thus, after reading $n-1$ unary symbols a we reach the state $(\{0, n-1\}, n-1)$.

Inductively, we assume that a state $(\{0, 1, 2, \ldots, k, n-1\}, n-1)$, $0 \leq k < n-2$, is reachable. We show that $(\{0, 1, 2, \ldots, k+1, n-1\}, n-1)$ is also reachable. From state $(\{0, 1, 2, \ldots, k, n-1\}, n-1)$, we reach the state $Z_1 = (\{1, 2, \ldots, k+1, 0\}, 0)$ by reading a unary symbol a. By our assumption on k, $k + 1 < n - 1$. Thus from Z_1 we reach the state $Z_2 = (\{2, 3, \ldots, k+2, 0\}, 0)$ by reading b. Since $k < n-2$, all elements of $\{2, 3, \ldots, k+2, 0\}$ are distinct (that is, the b-transition does not take $k+1$ to 0). After reading $n-1$ symbols a, the state $(\{1, 2, \ldots, k+1, n-1, 0\}, n-1)$ is reached. The element 0 is added to the first component as the second component becomes $n - 1$.

By the above inductive claim we now know that the state $(\{0, 1, \ldots, n-2, n-1\}, n-1)$ is reachable. After reading $i + 1$ $a's$, state $(\{0, 1, \ldots, n-2, n-1\}, i)$ is reached, $0 \leq i \leq n-1$.

Inductively, assume that all states (S, j), where $|S| \geq k + 1$, $1 \leq k < n$ and $0 \leq j \leq n - 1$ as in (5) are reachable. We show that then also states where

$|S| = k$ are reachable. Let (S, s_i) where $S = \{s_1, s_2, \ldots, s_k\}$, $1 \le i \le k$ and $0 \le s_1 < s_2 < \ldots < s_k \le n-1$ be an arbitrary state where $|S| = k$. Recall that in states of M_B, when the second component is not n, it must belong to the first component.

In the below cases (a) and (b), numbers $z \ge n$ are interpreted as the unique element of $\{0, 1, \ldots, n-1\}$ congruent to z modulo n.

(a-i) First consider the case where $s_i < n-1$. The following discussion assumes $n \ge 3$, and the case $n = 2$ is handled in case (a-ii). Since $|S| = k < n$, in the "cyclical sequence" of $s_1, \ldots, s_k$, there exist two consecutive numbers with difference at least two, where the difference between the numbers s_k and s_1 is counted modulo n. More formally, either there exists $1 \le j \le k-1$ such that $s_{j+1} - s_j \ge 2$ or $n + s_1 - s_k \ge 2$. In the latter case we choose $j = k$. In the following we assume that $i \le j$. The case where $i > j$ is similar and only some notations are changed. According to the inductive assumption, the state $Z_3 = (\{0, n-1\} \cup S_1, n + s_i - s_j - 1)$ where $S_1 = \{s_{j+1} - s_j - 1, s_{j+2} - s_j - 1, \ldots, s_k - s_j - 1, n + s_1 - s_j - 1, n + s_2 - s_j - 1, \ldots, n + s_{j-1} - s_j - 1\}$ is reachable. Note that since $0 \le s_1 < s_2 < \ldots < s_k \le n-1$ and $s_{j+1} - s_j \ge 2$, $|S_1 \cup \{0, n-1\}| = k+1$. After reading from state Z_3 a unary symbol b, we get the state $Z_4 = (\{0\} \cup S_2, n + s_i - s_j)$ where $S_2 = \{s_{j+1} - s_j, s_{j+2} - s_j, \ldots, s_k - s_j, n + s_1 - s_j, n + s_2 - s_j, \ldots, n + s_{j-1} - s_j\}$. Since $0 \le s_1 < s_2 < \ldots < s_k \le n-1$, $0 \notin S_2$. From state Z_4 we reach the state $(\{s_j, s_{j+1}, s_{j+2}, \ldots, s_k, n + s_1, n + s_2, \ldots, n + s_{j-1}\}, n + s_i)$ by reading s_j symbols a. The latter state is the state (S, s_i) that we wanted.

(a-ii) Assume that $s_i < n-1$ and $n = 2$. Now $k = 1$, and the only legal state (S, s_i), $|S| = k = 1$, $0 \le s_i < 1$, is $(\{0\}, 0)$ (because we know that $s_i \in S$). The state $(\{0\}, 0)$ is reached from state p_{new} by reading unary symbols ab.

(b) Now consider the case where $s_i = n-1$, and thus $i = k$. This implies that $0 \in S$, and we have $s_i (= s_k) = n-1$ and $s_1 = 0$. Since $k < n$, there exists $1 \le j \le k-1$ such that $s_{j+1} - s_j \ge 2$. According to the inductive assumption, the state $Z_5 = (\{0, n-1\} \cup S_3, n - 2 - s_j)$ is reachable, where $S_3 = \{s_{j+1} - s_j - 1, s_{j+2} - s_j - 1, \ldots, s_{k-1} - s_j - 1, n - 1 - s_j - 1, n + 0 - s_j - 1, n + s_2 - s_j - 1, \ldots, n + s_{j-1} - s_j - 1\}$. Similarly as in (a) above we observe that $|S_3 \cup \{0, n-1\}| = k+1$. From state Z_5 we get the state $Z_6 = (\{s_{j+1} - s_j, s_{j+2} - s_j, \ldots, s_{k-1} - s_j, n - 1 - s_j, n + 0 - s_j, n + s_2 - s_j, \ldots, n + s_{j-1} - s_j, 0\}, n - 1 - s_j)$ by reading a symbol b. After reading s_j symbols a, from state Z_6 we reach the state $(\{s_{j+1}, s_{j+2}, \ldots, s_{k-1}, n-1, n+0, n+s_2, \ldots, n+s_{j-1}, s_j\}, n-1)$. This means that we have reached the desired state $(S, n-1)$ with $S = \{0, s_2, \ldots, s_{k-1}, n-1\}$.

Up to now, we have shown that all that states (S, j), $S \subseteq \{0, \ldots, n-1\}$, $0 \le j \le n-1$ as in (5) are reachable. Next we will show that the states (S, n), $S \subset \{0, 1, \ldots, n-1\}$ are reachable.

We know that $(\{0, 1, \ldots, n-1\}, 0)$ is reachable and from this state we get $Z_7 = (\{1, \ldots, n-1\}, n)$ by reading a unary symbol c. From Z_7 we get all states

(S, n), $|S| = n-1$ by cycling the elements of S using a-transitions. Now inductively, assume that all states (S, n), $n > |S| \geq k+1$, $k < n-1$ are reachable. Consider an arbitrary state (S, n) where $|S| = k$. Choose $0 \leq j \leq n-1$ such that $j \notin S$. By our inductive assumption the state $(S \cup \{j\}, n)$ is reachable. From this state we reach (S, n) by reading the sequence of unary symbols $a^{n-j}ca^{j}$. Note that transitions on a always add one modulo n to states of S and the c-transition deletes the element 0 and is the identity on all other elements.

It remains to consider the state $(\{0, 1, \ldots, n-1\}, n)$. We know that states $(\{0, 1\}, 0)$ and $(\{0, 1, \ldots, n-1\}, 1)$ are reachable. According to the definition of d_2-transitions of M_A, the d_2-transition of M_B with arguments $(\{0, 1\}, 0)$ and $(\{0, 1, \ldots, n-1\}, 1)$ gives the state $(\{0, 1, \ldots, n-1\}, n)$. □

Note that above the transitions on d_2 were needed only to establish that the state $(\{0, 1, \ldots, n-1\}, n)$ is reachable in M_B. The transitions of d_2 in M_A did not have a similar intuitive interpretation as the other transitions based on the DFA A, and they were introduced only for the technical purpose needed at the end of the proof of Lemma 4.

By Lemmas 2, 3 and 4 we have a tight bound for the state complexity of bottom-up star that differs by an order of magnitude from the known bound for Kleene-star of string languages [4, 24].

Theorem 1. *If A is a DTA with n states, the bottom-up star of $L(A)$ can be recognized by a DTA with $(n + \frac{3}{2}) \cdot 2^{n-1}$ states. For every $n \geq 2$, there exists an n-state DTA A and $\sigma \in \Sigma_0$ such that the minimal DTA for $L(A)^{b,*}_{\sigma}$ has $(n + \frac{3}{2}) \cdot 2^{n-1}$ states.*

Next we give a tight state complexity bound for top-down star of regular tree languages. The top-down iteration of the concatenation operation allows the replacement of subtrees at arbitrary locations and, as can perhaps be expected, the state complexity is similar as for the Kleene-star of string languages. However, it should be noted that we are considering incomplete automata and the known state complexity bounds for ordinary DFAs are stated in terms of complete DFAs [24, 25]. The state complexity results for complete and incomplete DFAs, respectively, differ slightly for operations such as union or concatenation [24, 18].

Theorem 2. *Let $A = (\Sigma, Q_A, Q_{A,F}, g_A)$ be a DTA with n states and $\sigma \in \Sigma_0$. The top-down σ-star of the tree language recognized by A, $L(A)^{t,*}_{\sigma}$, can be recognized by a DTA B with $\frac{3}{4} \cdot 2^n$ states and this bound can be reached in the worst case.*

Proof. The construction of $B = (\Sigma, Q_B, Q_{B,F}, g_B)$ is similar as the construction used to recognize the Kleene-star of a string language. The set of states Q_B consists of nonempty subsets of $P \subseteq Q_A$ such that $P \cap Q_{A,F} \neq \emptyset$ implies $\sigma_{g_A} \in P$, and additionally Q_B has one new state q_{new} that is reached at leaves labeled by σ (the symbol that defines the star operation). Note that the state q_{new} is used as a copy of σ_{g_A} because the latter state is not, in general, accepting. The cardinality of Q_B is maximized as $2^n - 1 - 2^{n-2} + 1 = \frac{3}{4} \cdot 2^n$ by choosing $|Q_{A,F}| = 1$. We leave details of the construction to the reader.

When restricted to unary trees, the top-down (or bottom) star operation coincides with Kleene-star on string languages. Theorem 5.5 of [24] gives a complete DFA C with n states such that the state complexity of the Kleene-star of $L(C)$ is $\frac{3}{4} \cdot 2^n$. Furthermore, C does not have a dead state, which means that the same lower bound construction works for incomplete DFAs. □

4 Kleene-Star Combined with Concatenation

The worst case state complexity of star–of–concatenation of string languages is known [2]. However, already in the case of string languages determining the precise state complexity of combined operations is often quite involved [2, 10].

For tree languages we consider a restricted case of Kleene-star combined with concatenation where one of the arguments for concatenation is the set of all trees F_Σ. For some of the combined operations we get tight bounds that are significantly lower than the function composition of the state complexity of the individual operations. Altogether there are four combinations of bottom-up star (or top-down star) with the parallel or sequential concatenation with the set of all trees. The combined operations for bottom-up star are as follows:

$$(F_\Sigma \cdot_\sigma^p L)_\sigma^{b,*}, \ \ (L \cdot_\sigma^p F_\Sigma)_\sigma^{b,*}, \ \ (L \cdot_\sigma^s F_\Sigma)_\sigma^{b,*}, \text{ and } (F_\Sigma \cdot_\sigma^s L)_\sigma^{b,*}.$$

It turns out that, for the first and the last of the listed combined operations, the tree automaton constructions can be significantly simplified by relying on general observations about the (parallel or sequential) concatenation of a general tree language with the set of all trees.

Lemma 5. *Let $L \subseteq F_\Sigma$ and $\sigma \in \Sigma_0$. Then*

(i) $(F_\Sigma \cdot_\sigma^p L)_\sigma^{b,*} = (F_\Sigma \cdot_\sigma^p L)_\sigma^{t,*} = F_\Sigma \cdot_\sigma^p L \cup \{\sigma\}$,

(ii) $(L \cdot_\sigma^s F_\Sigma)_\sigma^{b,*} = (L \cdot_\sigma^s F_\Sigma)_\sigma^{t,*} = L \cdot_\sigma^s F_\Sigma \cup \{\sigma\}$.

Using Lemma 5, we get tight state complexity bounds for two combined operations involving bottom-up star and top-down star, respectively.

Theorem 3. *Let A be a DTA with n states and $\sigma \in \Sigma_0$. Then, $(F_\Sigma \cdot_\sigma^p L(A))_\sigma^{b,*}$ can be recognized by a DTA with $2^{n-1} + 1$ states and this bound can be reached in the worst case.*

Proof. Let $A = (\Sigma, Q_A, Q_{A,F}, g_A)$ be a DTA with n states recognizing the tree language L. Without loss of generality we assume that σ_{g_A} is defined, because otherwise $F_\Sigma \cdot_\sigma^p L(A) = L(A)$, and $(F_\Sigma \cdot_\sigma^p L(A))_\sigma^{b,*} = L(A) \cup \{\sigma\}$ and we can easily construct a DTA with $n+1$ states to recognize $L(A) \cup \{\sigma\}$.

We define a DTA $B = (\Sigma, Q_B, Q_{B,F}, g_B)$ where

$$Q_B = 2^{Q_A} \cup \{q_{\text{new}}\}, \ \ Q_{B,F} = \{P \in Q_B \mid P \cap Q_{A,F} \neq \emptyset\} \cup \{q_{\text{new}}\},$$

and the transitions of g_B are defined as below. Note that q_{new} can be viewed as a copy of the state $\{\sigma_{g_A}\}$. The reason why we have an additional state q_{new} is because q_{new} needs to be an accepting state and $\{\sigma_{g_A}\}$ is not accepting, in general.

For $\tau \in \Sigma_0$, $\tau \neq \sigma$, $\tau_{g_B} = \{\tau_{g_A}, \sigma_{g_A}\}$, and, $\sigma_{g_B} = q_{\text{new}}$. For $P \in Q_B$, define $\overline{P} \subseteq Q_A$ by

$$\overline{P} = \begin{cases} P \text{ if } P \in 2^{Q_A}, \\ \{\sigma_{g_A}\} \text{ if } P = q_{\text{new}}. \end{cases}$$

Now for $\tau \in \Sigma_k, k \geq 1$, and $P_i \in Q_B, i = 1, \ldots, k$, define

$$\tau_{g_B}(P_1, \ldots, P_k) = \tau_{g_A}(P_1, \ldots, P_k) \cup \{\sigma_{g_A}\}.$$

We leave to the reader the details of verifying that B recognises the tree language $F_\Sigma \cdot_\sigma^p L(A) \cup \{\sigma\}$. Among the states $P \in Q_B$, the sets where $\sigma_{g_A} \notin P$ are unreachable. Therefore, the number of reachable states of B is at most $2^{n-1} + 1$.

For the lower bound, we can modify the corresponding construction by Yu et al. [25] for string languages. The proof of Theorem 2.1 of [25] gives an n-state DFA C[3] over alphabet $\Gamma = \{a, b\}$ such that

$$\Gamma^* \cdot L(C) = \{w \in \Gamma^* \mid w = ubv, |v|_a \equiv n-2 \pmod{n-1}\}.$$

and verifies that the state complexity of $\Gamma^* \cdot L(C)$ is 2^{n-1}. We note that the empty string is not in $\Gamma^* \cdot L(C)$. Thus, when C is interpreted as a tree automaton C' with unary symbols a, b and a nullary symbol σ, a tree automaton recognizing $F_\Sigma \cdot_\sigma^p L(C') \cup \{\sigma\}$ needs one additional state for the leaf symbol σ. □

Now by Lemma 5 (i) and Theorem 3 we get a tight state complexity bound for the corresponding combined operation involving top-down star.

Corollary 1. *If $L \subset F_\Sigma$ is recognized by a DTA with n states, for any $\sigma \in \Sigma_0$, the tree language $(F_\Sigma \cdot_\sigma^p L)_\sigma^{t,*}$ has a DTA with $2^{n-1}+1$ states and this number of states is necessary in the worst case.*

Theorem 4. *Let A be a DTA with n states and $\sigma \in \Sigma_0$. Then, $(L(A) \cdot_\sigma^s F_\Sigma)_\sigma^{b,*}$ can be recognized by a DTA with $n+2$ states and this bound can be reached in the worst case.*

Proof. Let $A = (\Sigma, Q_A, Q_{A,F}, g_A)$ be a DTA with n states recognizing the tree language L. We define a DTA $B = (\Sigma, Q_B, Q_{B,F}, g_B)$ for the tree language $(L(A) \cdot_\sigma^s F_\Sigma)_\sigma^{b,*} = L(A) \cdot_\sigma^s F_\Sigma \cup \{\sigma\}$. The following construction assumes that σ_{g_A} is defined and $\sigma_{g_A} \notin Q_{A,F}$. If either of these two conditions is not satisfied, the construction is similar and simpler (in both cases B can do with one fewer state).

Choose

$$Q_B = Q_A \cup \{q_\sigma, q_{\text{dummy}}\}, \quad Q_{B,F} = Q_{A,F} \cup \{q_\sigma\},$$

[3]In the notations of [25], the DFA C is called B.

and the transitions of g_B are defined as below. For $\tau \in \Sigma_0$,

$$\tau_{g_B} = \begin{cases} q_\sigma & \text{if } \tau = \sigma, \\ \tau_{g_A} & \text{if } \tau \neq \sigma \text{ and } \tau_{g_A} \text{ is defined,} \\ q_{\text{dummy}} & \text{otherwise.} \end{cases}$$

Define $g : Q_B \to Q_B$ by setting $g(q_\sigma) = \sigma_{g_A}$ and $g(q) = q$ when $q \neq q_\sigma$. Recall that we assumed that σ_{g_A} is defined. Let q_{final} be an arbitrary but fixed element of $Q_{A,F}$. Now for $\tau \in \Sigma_k, k \geq 1$, and $p_i \in Q_B, i = 1, \ldots, k$, define

$$\tau_{g_B}(p_1, \ldots, p_k) = \begin{cases} q_{\text{final}} & \text{if } \exists j, 1 \leq j \leq k \text{ where } p_j \in Q_{A,F}, \\ \tau_{g_A}(f(p_1), \ldots, f(p_k)) & \text{if } p_1, \ldots p_k \in (Q_A - Q_{A,F}) \cup \{q_\sigma\} \\ & \text{and } \tau_{g_A}(f(p_1), \ldots, f(p_k)) \text{ is defined,} \\ q_{\text{dummy}} & \text{in all other cases.} \end{cases}$$

The DTA B simulates the computation of A up to a point when it reaches a final state, and having reached a final state is marked by entering the state q_{final}. The state q_σ is entered only in a leaf labeled by σ and for transitions on symbols of Σ_k, $k \geq 1$, q_σ is treated as σ_{g_A}. The "copy" of the state σ_{g_A} is needed because B has to accept σ and σ_{g_A} is not accepting. If the computation of A reaches an undefined transition (before entering a final state), B enters the state q_{dummy}. Thus it is clear that B recognizes the set trees having a subtree in $L(A)$ and additionally the tree consisting of the single leaf labeled by σ.

Next we show that the upper bound $n + 2$ is tight. Choose $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$, where $\Sigma_0 = \{c\}$, $\Sigma_1 = \{a\}$ and $\Sigma_2 = \{b\}$. We define a DTA $C = (\Sigma, Q_C, Q_{C,F}, g_C)$, where $Q_C = \{0, 1, \ldots, n-1\}, Q_{C,F} = \{n-1\}$, and the transition function g_C is defined by setting:

$$c_{g_C} = 0, \quad a_{g_C}(i) = i + 1 \pmod{n} \quad \text{for } 0 \leq i \leq n-1.$$

All transitions not listed above are undefined. In particular, note that all transitions for the binary symbol b are undefined. Based on C, we construct a DTA $D = (\Sigma, Q_D, Q_{D,F}, g_D)$ recognizing $(L(A) \cdot_c^s F_\Sigma)_c^{b,*} = L(A) \cdot_c^s F_\Sigma \cup \{c\}$, as described above. Here $Q_D = Q_C \cup \{q_c, q_{\text{dummy}}\}$, $Q_{D,F} = Q_{C,F} \cup \{q_c\}$.

We verify that all states of D are reachable and pairwise inequivalent, and none of the states is a dead state. The state 1 is reached by reading the tree $a(c)$. Then the cyclic transitions on unary symbols a guarantee that states $2, 3, \ldots n$ and 0 are also reachable. The state q_c is reached in a leaf labeled by c and q_{dummy} is reachable because C has undefined transitions.

States $0 \leq i < j \leq n-1$ are not equivalent because by reading $n - 1 - i$ unary symbols a the state i ends in the accepting state $n - 1$ and by reading the same sequence of unary symbols j does not enter an accepting state. By the same reasoning q_c is not equivalent to any state $1 \leq j \leq n$. The state q_c is not equivalent with 0 because the former is a final state and the latter is not. The state q_{dummy} cannot reach a final by reading a sequence of a's while all other states have this property. Finally to verify that none of the states is a dead state, above we have already observed that the states $0 \le i \le n-1$ and q_c can reach a final state by reading a sequence of a's. According to the definition of the transitions of D, $b_{g_D}(q_{\text{dummy}}, n-1) = n-1$ and it follows that also q_{dummy} is not a dead state. (Note that in the DTA D we must have $q_{\text{final}} = n-1$ since $n-1$ is the only final state of C.)

We have verified that the minimal DTA for $L(A) \cdot_c^s F_\Sigma \cup \{c\}$ has $n+2$ states and this concludes the proof. $\square$

In the construction used for the lower bound of Theorem 4, the symbol b of rank two has no defined transitions in the original DTA C. However, it can be noted that the tight bound cannot be reached by tree languages over a ranked alphabet that has no symbols of rank greater than one. If the ranked alphabet has only unary and nullary symbols, in the DTA B constructed to recognize the tree language $(L(A) \cdot_\sigma^s F_\Sigma)_\sigma^{b,*}$ the state q_{dummy} will always be a dead state.

Again using Lemma 5 (ii) and Theorem 4 we get a tight bound for the same combined operation involving top-down star:

Corollary 2. *For a tree language L recognized by a DTA with n states and $\sigma \in \Sigma_0$, the tree language $(L \cdot_\sigma^s F_\Sigma)_\sigma^{t,*}$ has a DTA with $n+2$ states and $n+2$ states is needed in the worst case.*

For establishing an upper bound for the combined operations $(L \cdot_\sigma^p F_\Sigma)_\sigma^{b,*}$ and $(L \cdot_\sigma^p F_\Sigma)_\sigma^{t,*}$ we first consider a construction for the parallel concatenation of L and F_Σ. If A is an n-state DFA on strings over alphabet Γ, the language $L(A) \cdot \Gamma^*$ can be recognized by a DFA with n states. For the parallel concatenation of an n-state tree language and F_Σ we use $2n$ states.

Lemma 6. *Let A be a DTA with n states and f final states and $\sigma \in \Sigma_0$. Then, $L(A) \cdot_\sigma^p F_\Sigma$ can be recognized by a DTA with $2n+1-f$ states.*

Proof. Let $A = (\Sigma, Q_A, Q_{A,F}, g_A)$. We construct a DTA $B = (\Sigma, Q_B, Q_{B,F}, g_B)$ for the tree language $L(A) \cdot_\sigma^p F_\Sigma$. Note that if $\sigma \in L(A)$, then $L(A) \cdot_\sigma^p F_\Sigma = F_\Sigma$. Without loss of generality we can assume that $\sigma_{g_A} \notin Q_{A,F}$. Choose

$$Q_B = \{0,1\} \times (Q_A - Q_{A,F}) \;\cup\; \{0\} \times (Q_{A,F} \cup \{q_{\text{Adead}}\}),$$

where q_{Adead} is a new element not in Q_A, $Q_{B,F} = \{(0,q) \mid q \in Q_A \cup \{q_{\text{Adead}}\}\}$ and the transitions of g_B are defined as below. We set $\sigma_{g_B} = (1, \sigma_{g_A})$ if σ_{g_A} is defined, and σ_{g_B} is undefined otherwise. For $\tau \in \Sigma_0$, $\tau \neq \sigma$,

$$\tau_{g_B} = \begin{cases} (0, \tau_{g_A}) & \text{if } \tau_{g_A} \text{ is defined,} \\ (0, q_{\text{Adead}}) & \text{if } \tau_{g_A} \text{ is not defined.} \end{cases}$$

For $\tau \in \Sigma_k, k \ge 1$, and $x_1, \ldots, x_k \in \{0,1\}$, $q_1, \ldots, q_k \in Q_A \cup \{q_{\text{Adead}}\}$ we define $\tau_{g_B}((x_1,q_1), \ldots, (x_k,q_k))$ to be

(i) $(1, \tau_{g_A}(q_1, \ldots, q_k))$ if there exists $1 \le i \le k$ such that $x_i = 1$ and $\tau_{g_A}(q_1, \ldots, q_k) \in Q - Q_{A,F}$,

(ii) $(0, \tau_{g_A}(q_1, \ldots, q_k))$ if $\tau_{g_A}(q_1, \ldots, q_k) \in Q_{A,F}$,

(iii) $(0, \tau_{g_A}(q_1, \ldots, q_k))$ if $x_i = 0$, $i = 1, \ldots, k$ and $\tau_{g_A}(q_1, \ldots, q_k))$ is defined,

(iv) $(0, q_{\text{Adead}})$ if $\tau_{g_A}(q_1, \ldots, q_k)$ is undefined and $x_1 = \ldots = x_k = 0$,

(v) undefined in all the remaining cases.

Note that in the above definitions if some q_i is q_{Adead}, the transition $\tau_{g_A}(q_1, \ldots, q_k)$ is naturally undefined.

We note that the tree language $L(A) \cdot_\sigma^p F_\Sigma$ consists of all Σ-trees t that have the property that any leaf labeled by σ must belong to a subtree of t that is in $L(A)$. The DTA B checks this property as follows. The second components of the states simulate the computation of A and the bit in the first component keeps track of whether or not the current subtree has a leaf labeled by σ that "was not part of a subtree" belonging to $L(A)$. More precisely, suppose that the computation reaches the root of t in state (x, q). If $x = 1$, this indicates t had a leaf ℓ labeled by σ and the computation from ℓ to the root of t has not passed through a final state of A. Note that in the transitions of B when the second component becomes an element of $Q_{A,F}$ the first component is always reset to 0, that is, pairs of the form $(1, q)$, $q \in Q_{A,F}$, are not used as states of B. If the second component of the state is q_{Adead}, this indicates that the computation of A on the current subtree t is undefined. In this situation if t contains a leaf labeled by σ, t cannot be a subtree of $L(A) \cdot_\sigma^p F_\Sigma$ and the state $(1, q_{\text{Adead}})$ is not in Q_B.

The claim follows since $|Q_B| = 2 \cdot |Q_A| + 1 - |Q_{A,F}|$. □

Now combining Lemma 6 with, respectively, Theorem 1 and Theorem 2 we get the following upper bounds for the state complexity of bottom-up or top-down star of a tree language $L \cdot_\sigma^p F_\Sigma$. Note that the bound of Lemma 6 reaches the worst case $2n$ when the DTA has exactly one final state.

Proposition 1. *Let A be a DTA with n states and $\sigma \in \Sigma_0$. Then $(L(A) \cdot_\sigma^p F_\Sigma)_\sigma^{b,*}$ can be recognized by a DTA with $(4n + 3) \cdot 4^{n-1}$ states.*

The tree language $(L(A) \cdot_\sigma^p F_\Sigma)_\sigma^{t,*}$ can be recognized by a DTA with $3 \cdot 4^{n-1}$ states.

We do not know whether the bounds of Proposition 1 are optimal. Finally, the bottom-up or top-down star of the sequential concatenation of F_Σ with a regular tree language L, $F_\Sigma \cdot_\sigma^s L$, seem to be the most problematic of the combined operations involving Kleene star and concatenation with F_Σ. For these operations we know only trivial upper bounds implied by the state complexity of sequential concatenation and the corresponding star operation.

5 Conclusion

In the last section we have considered the state complexity of star–of–concatenation in the special case where one of the argument tree languages consists of the set of all trees. The precise state complexity of star–of–concatenation remains open for general tree languages. For references dealing with the string case the reader may consult [2].

For top-down and bottom-up star we have established the precise worst case state complexity. The lower bound construction for Kleene-star in [24] uses a two-letter alphabet, and hence the worst-case state complexity of top-down star can be achieved over a ranked alphabet with two unary and one nullary symbol. It is clear that one unary and one nullary symbol is not sufficient because the state complexity of Kleene-star for string languages over a one-letter alphabet is $(n-1)^2+1$ [24]. With one binary symbol ω and one nullary symbol σ, we can encode strings over a two letter alphabet as trees "built up" from elements $\omega(\sigma, x)$ and $\omega(x, \sigma)$. In this way one clearly gets an exponential lower bound construction, however, we do not know whether one binary and one nullary symbol is sufficient to reach the precise bound of Theorem 2.

Our lower bound construction for Theorem 1 uses a ranked alphabet of six symbols. The state complexity for bottom-up star is of a different order of magnitude than the corresponding bound for string languages. This means that the worst-case constructions essentially need to rely on "tree properties" and finding the minimal alphabet size remains an open question.

References

[1] Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Löding, C., Tison, S., Tommasi, M.: *Tree Automata Techniques and Applications,* electronic book available at `tata.gforge.inria.fr`, 2007.

[2] Cui, B., Gao, Y., Kari, L., Yu, S.: State complexity of combined operations with two basic operations. Theoret. Comput. Sci. 437 (2012) 82–102.

[3] Eom, H.-S., Han, Y.-S., Ko, S.-K.: State complexity of subtree free regular tree languages, *Proc. DCFS'15*, LNCS 8031, Springer, 2013, pp. 66–77.

[4] Gao, Y., Moreira, N., Reis, R., Yu, S.: A review on state complexity of individual operations. To appear in *Computer Science Review.* (Available at `www.dcc.fc.up.pt/dcc/Pubs/TReports/TR11/dcc-2011-08.pdf`)

[5] Gécseg, F., Steinby, M.: *Tree automata,* Akadémiai Kiadó, 1984.

[6] Gécseg, F., Steinby, M.: Tree languages, *Handbook of Formal Languages,* Vol. III, (G. Rozenberg, A. Salomaa Eds.), Springer, 1997, pp. 1–68.

[7] Han, Y.-S., Salomaa, K.: Nondeterministic state complexity of nested word automata. Theoret. Comput. Sci. 410 (2009) 2961–2971.

[8] Holzer, M., Kutrib, M.: Descriptional and computational complexity of finite automata — A survey. Inf. Comput. 209 (2011) 456–470.

[9] Holzer, M., Salomaa, K., Yu, S.: On the state complexity of k-entry deterministic finite automata. J. Autom., Lang. and Combinatorics 6 (2001) 453–466.

[10] Jirásková, G., Okhotin, A.: On the state complexity of star of union and star of intersection. Fund. Informaticae 109 (2011) 161–178.

[11] Ko, S.-K., Lee, H.-R., Han, Y.-S.: State complexity of regular tree languages for tree pattern matching, *Proc. of DCFS'14*, LNCS 8614, Springer, 2014, pp. 246–257.

[12] Kutrib, M., Pighizzini, G.: Recent trends in descriptional complexity of formal languages. *Bulletin of the EATCS* 111 (2013) 70–86.

[13] Lupanov, O.B.: A comparison of two types of finite sources. *Problemy Kibernetiki* 9 (1963) 328–335.

[14] Martens, W., Niehren, J.: On the minimization of XML schemas and tree automata for unranked trees. J. Comput. System Sci. 73 (2007) 550–583.

[15] Maslov, A.N.: Estimates of the number of states of finite automata, *Dokl. Akad. Nauk. SSSR,* **194** (1970) Soviet Math. Dokl. 11 (1970) 1373–1375.

[16] Meyer, A.R., Fischer, M.J.: Economy of description by automata, grammars and formal systems. Proc. SWAT (FOCS), IEEE Computer Society (1971) 188–191.

[17] Okhotin, A., Salomaa, K.: Descriptional complexity of unambiguous input-driven pushdown automata. Theoret. Comput. Sci. 566 (2015) 1–11.

[18] Piao, X., Salomaa, K.: State complexity of the concatenation of regular tree languages. Theoret. Comput. Sci. 429 (2012) 273–281

[19] Piao, X., Salomaa, K.: Transformations between different models of unranked bottom-up tree automata. Fund. Informaticae 109 (2011) 405–424.

[20] Piao, X., Salomaa, K.: State complexity of star and quotient operation for unranked tree automata, School of Computing, Queen's University Technical Report No. 2011-577 (19 pp.), 2011

[21] Salomaa, A., Salomaa, K., Yu, S.: State complexity of combined operations. Theoret. Comput. Sci. 383 (2007) 140–152.

[22] Schwentick, T.: Automata for XML, — a survey. J. Comput. System Sci. 73 (2007) 289–315.

[23] Shallit, J.: *A Second Course in Formal Languages and Automata Theory,* Cambridge University Press, 2009.

[24] Yu, S.: Regular languages, in: *Handbook of Formal Languages,* Vol. I, (G. Rozenberg, A. Salomaa, Eds.), Springer, 1997, pp. 41–110.

[25] Yu, S., Zhuang, Q., Salomaa, K.: The state complexity of some basic operations on regular languages. Theoret. Comput. Sci. 125 (1994) 315–328.

Received 2nd June 2015