# State Complexity of Regular Tree Languages for Tree Pattern Matching

Sang-Ki Ko, Ha-Rim Lee, and Yo-Sub Han

Department of Computer Science, Yonsei University,
50, Yonsei-Ro, Seodaemun-Gu, Seoul 120-749, Korea
{narame7,hrlee,emmous}@cs.yonsei.ac.kr

**Abstract.** We study the state complexity of regular tree languages for tree matching problem. Given a tree $t$ and a set of pattern trees $L$, we can decide whether or not there exists a subtree occurrence of trees in $L$ from the tree $t$ by considering the new language $L'$ which accepts all trees containing trees in $L$ as subtrees. We consider the case when we are given a set of pattern trees as a regular tree language and investigate the state complexity. Based on the sequential and parallel tree concatenation, we define three types of tree languages for deciding the existence of different types of subtree occurrences. We also study the deterministic top-down state complexity of path-closed languages for the same problem.

**Keywords:** tree automata, state complexity, tree pattern matching, regular tree languages.

## 1 Introduction

State complexity is one of the most interesting topics in automata and formal language theory [6,7,18,19]. The state complexity of finite automata has been studied since the 60's [8,10,11]. Maslov [9] initiated the problem of finding the operational state complexity and Yu et al. [19] investigated the state complexity for basic operations.

Recently, the state complexity problem has been extended to regular tree languages. Regular tree languages and tree automata theory provide a formal framework for XML schema languages such as XML DTD, XML Schema, and Relax NG [12]. XML schema languages can process a set of XML documents by specifying the structural properties. Piao and Salomaa [14,15] considered the state complexity between different models of unranked tree automata. They also investigated the state complexity of concatenation [17] and star [16] for regular tree languages. Two of the authors studied the state complexity of subtree-free regular tree languages, which are a proper subclass of regular tree languages [3].

Since a regular tree language is a set of trees, it is suitable for representing a set of structural documents such as XML documents, web documents, or RNA secondary structures. This implies that a regular tree language can be used as a theoretical toolbox for processing of the structured documents. When it comes to the string case, many researchers often use regular languages to process a set of strings efficiently. Consider the case that we have a set of strings which is a regular language $L$. Now we want to find any occurrence of strings in $L$ from a text $T$. The most common way is to construct an FA $A$ that accepts a regular language $\Sigma^*L$ [2]. Then, we read $T$ using $A$ and check whether or not $A$ reaches a final state. When $A$ reaches a final state, we find that there is an occurrence of a matching string of $L$ in $T$. We extend this approach to the tree matching problem [5]. First, we formally define the *tree matching problem* to be the problem of finding subtree occurrences of a tree in $L$ from a set of trees $T$. Since a tree can be processed in a bottom-up or a top-down fashion, we need to consider different types of tree languages for the tree matching problem.

Here we consider three types of tree substructures called a *subtree*, a *topmost subtree* and an *internal subtree*. Given a tree language $L$, we construct three types of tree languages recognizing trees which contain the trees in $L$ as subtrees, topmost subtrees and internal subtrees. Note that these tree languages can be used for the tree matching problem as we have used $\Sigma^*L$ for the string pattern matching problem. In particular, we tackle the deterministic state complexity of regular tree languages and path-closed languages. Interestingly, the tree language consisting of trees that have a subtree belonging to a path-closed language language need not be path-closed and therefore cannot recognized by deterministic top-down tree automata (DTTAs).

We give basic notations and definitions in Section 2. We define the three types of tree languages for tree matching in Section 3. We present the results on the state complexity of regular tree languages and path-closed languages in Section 4 and Section 5. In Section 6, we conclude the paper.

## 2 Preliminaries

We briefly recall definitions and properties of finite tree automata and regular tree languages. We refer the reader to the books [1,4] for more details on tree automata. A ranked alphabet $\Sigma$ is a finite set of characters and we denote the set of elements of rank $m$ by $\Sigma_m \subseteq \Sigma$ for $m \geq 0$. The set $F_\Sigma$ consists of $\Sigma$-labeled trees, where a node labeled by $\sigma \in \Sigma_m$ always has $m$ children. We use $F_\Sigma$ to denote a set of trees over $\Sigma$ that is the smallest set $S$ satisfying the following condition: if $m \geq 0, \sigma \in \Sigma_m$ and $t_1, \ldots, t_m \in S$, then $\sigma(t_1, \ldots, t_m) \in S$. Let $t(u \leftarrow s)$ be the tree obtained from a tree $t$ by replacing the subtree at a node $u$ of $t$ with a tree $s$. The notation is extended for a set $U$ of nodes of $t$ and $S \subseteq F_\Sigma : t(U \leftarrow S)$ is the set of trees obtained from $t$ by replacing the subtree at each node of $U$ by some tree in $S$.

A *nondeterministic bottom-up tree automaton* (NBTA) is specified by a tuple $A = (\Sigma, Q, Q_f, g)$, where $\Sigma$ is a ranked alphabet, $Q$ is a finite set of states, $Q_f \subseteq Q$ is a set of final states and $g$ associates each $\sigma \in \Sigma_m$ to a mapping $\sigma_g : Q^m \longrightarrow 2^Q$, where $m \geq 0$. For each tree $t = \sigma(t_1, \ldots, t_m) \in F_\Sigma$, we define inductively the set $t_g \subseteq Q$ by setting $q \in t_g$ if and only if there exist $q_i \in (t_i)_g$, for $1 \leq i \leq m$, such that $q \in \sigma_g(q_1, \ldots, q_m)$. Intuitively, $t_g$ consists of the states of $Q$ that $A$ may reach by reading $t$. Thus, the tree language accepted by $A$ is defined as follows: $L(A) = \{t \in F_\Sigma \mid t_g \cap Q_f \neq \emptyset\}$. The automaton $A$ is a *deterministic bottom-up tree automaton* (DBTA) if, for each $\sigma \in \Sigma_m$, where $m \geq 0$, $\sigma_g$ is a partial function $Q^m \longrightarrow Q$.

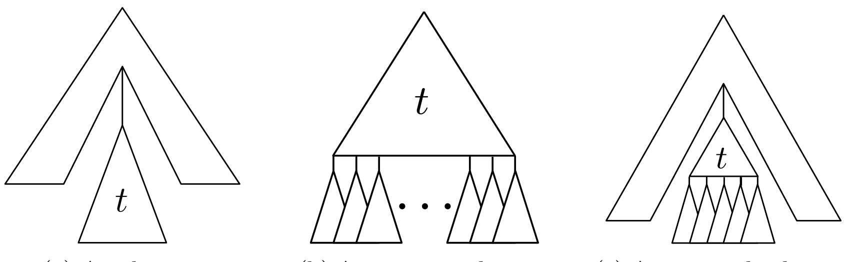
A *nondeterministic top-down tree automaton* (NTTA) is specified by a tuple $A = (\Sigma, Q, Q_0, g)$, where $\Sigma$ is a ranked alphabet, $Q$ is a finite set of states, $Q_0 \subseteq Q$ is a set of initial states, and $g$ associates each $\sigma \in \Sigma_m, m \geq 0$, a mapping $\sigma_g : Q \longrightarrow 2^{Q^m}$. As a convention, we denote the $m$-tuples $q_1, \ldots, q_m$ by $[q_1, \ldots, q_m]$. A top-down tree automaton $A$ is *deterministic* if $Q_0$ is a singleton set and for all $q \in Q, \sigma \in \Sigma_m$, and $m \geq 1$, $\sigma_g$ is a partial function $Q \longrightarrow Q^m$.

The nondeterministic (bottom-up or top-down) and deterministic bottom-up tree automata accept the family of *regular tree languages* whereas the deterministic top-down tree automata accept a proper subfamily of regular tree languages—*path-closed languages* [1,4].

## 3 Tree Languages for Tree Pattern Matching

Pattern matching is the problem of finding occurrences of a pattern in a text. Given an FA $A$ for the pattern $L$ over $\Sigma$, we can solve the problem by building a new FA for the language $\Sigma^*L$. Then, we run the new FA with the text and report the occurrence when the FA reaches a final state [2]. For tree pattern matching problem, we consider the case when we are given a set of pattern trees as a tree automaton. Note that a tree can be processed in a bottom-up way with a bottom-up TA or a top-down way with a top-down TA. Therefore, we consider three types of tree languages that can be used for tree pattern matching problem.

First we introduce our definitions for different tree substructures. We provide graphical examples for the definitions in Fig. 1.



(a) A subtree $t$ (b) A topmost subtree $t$ (c) An internal subtree $t$

**Fig. 1.** We define three types of subtrees called a subtree, a topmost subtree and an internal subtree. These figures depict the examples.

**Definition 1.** *A subtree of a tree $t$ is a tree consisting of a node in $t$ and all of its descendants in $t$.*

If a tree $t_1$ is a subtree of $t_2$, then we call $t_2$ is a supertree of $t_1$. Given a tree $t$ and a regular tree language $L$, we first compute a new regular tree language $L'$ that accepts all possible supertrees of trees in $L$. Then, we decide whether or not a given tree $t$ occurs as a subtree of a tree in $L$ by deciding $t \in L'$. Similarly, we define the *topmost subtree* and the *internal subtree* as follows:

**Definition 2.** *A topmost subtree of a tree $t$ is a tree consisting of a set of nodes in $t$ including the root node such that from any node in the set, there exists a path to the root node through the nodes in the set.*

**Definition 3.** *An internal subtree of a tree $t$ can be defined as a topmost subtree of a subtree of $t$.*

Recall that we build a new FA that accepts $\Sigma^* L$, which is a concatenation of a universal language $\Sigma^*$ and a given language $L$, for matching a language $L$ of string patterns. For tree pattern matching problem, we need to consider how to define the concatenation of trees properly. Recently, Piao and Salomaa [17] studied the state complexity of the concatenation of regular tree languages. They defined the sequential $\sigma$-concatenation and parallel $\sigma$-concatenation where the substitutions can occur at $\sigma$-labeled leaves.

We consider a more generalized operation that allows substitution to occur at all leaves regardless of labels. We denote the set of leaves of a tree $t$ by $\texttt{leaf}(t)$. Then, for $T_1 \subseteq F_\Sigma$ and $t_2 \in F_\Sigma$, we define the *sequential concatenation* of $T_1$ and $t_2$ to be $T_1 \cdot^s t_2 = \{t_2(u \leftarrow t_1) \mid u \in \texttt{leaf}(t_2), t_1 \in T_1\}$. In other words, $T_1 \cdot^s t_2$ is a set of trees obtained from $t_2$ by replacing a leaf with a tree in $T_1$. We extend the sequential concatenation operation to the tree languages $T_1, T_2 \subseteq F_\Sigma$ as follows:

$$T_1 \cdot^s T_2 = \bigcup_{t_2 \in T_2} T_1 \cdot^s t_2.$$

The *parallel concatenation* of $T_1$ and $t_2$ is

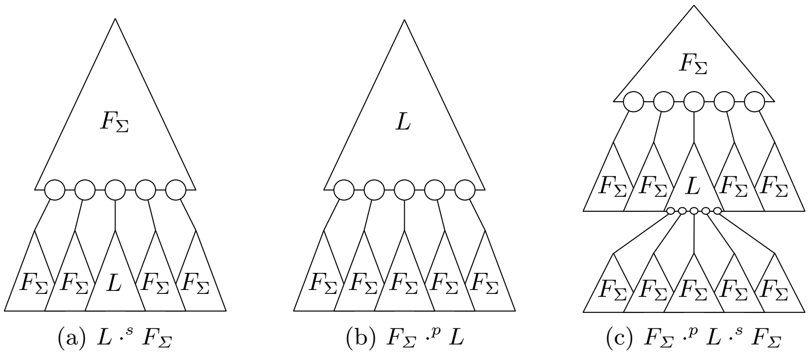
$$T_1 \cdot^p t_2 = \{t_2(\texttt{leaf}(t_2) \leftarrow t_1) \mid t_1 \in T_1\}.$$

Thus, $T_1 \cdot^p t_2$ is a set of trees obtained from $t_2$ by replacing all leaves with a tree in $T_1$. We can also extend the parallel concatenation to tree languages. Note that Definition 2 can be presented more nicely using the parallel concatenation operation. A tree $t_2$ is a topmost subtree of $t_1$ if $t_1 \in F_\Sigma \cdot^p t_2$.

Relying on the sequential and parallel tree concatenations, we construct three types of tree languages from a regular tree language $L$ for the tree pattern matching problem. See Fig. 2. Given a tree language $L$,

(i) $L \cdot^s F_\Sigma$ is a set of trees where each element contains a subtree occurrence of a tree in $L$,
(ii) $F_\Sigma \cdot^p L$ is a set of trees where each element contains a topmost subtree occurrence of a tree in $L$, and
(iii) $F_\Sigma \cdot^p L \cdot^s F_\Sigma$ contains trees having an internal subtree occurrence of a tree of $L$.

Notice that a leaf node of a tree can be replaced with any other nodes for the topmost subtree occurrence and the internal subtree occurrence.



$F_\Sigma$
$F_\Sigma$ $F_\Sigma$ $L$ $F_\Sigma$ $F_\Sigma$
(a) $L \cdot^s F_\Sigma$
$L$
$F_\Sigma$ $F_\Sigma$ $F_\Sigma$ $F_\Sigma$ $F_\Sigma$
(b) $F_\Sigma \cdot^p L$
$F_\Sigma$
$F_\Sigma$ $F_\Sigma$ $L$ $F_\Sigma$ $F_\Sigma$
$F_\Sigma$ $F_\Sigma$ $F_\Sigma$ $F_\Sigma$ $F_\Sigma$
(c) $F_\Sigma \cdot^p L \cdot^s F_\Sigma$

**Fig. 2.** Three types of tree languages for tree pattern matching problem

## 4 State Complexity of DBTAs

First we study the state complexity of $F_\Sigma \cdot^p L$ which can be used for finding subtree occurrences of a tree in $L$.

**Lemma 1.** *Given a DBTA $A = (\Sigma, Q, Q_F, g)$ with $n$ states for a regular tree language $L$, $2^{n-k}$ states are sufficient for recognizing $F_\Sigma \cdot^p L$ if $|\{\sigma_g \mid \sigma \in \Sigma_0\}| = k$.*

*Proof.* Without loss of generality, we assume $Q_F \cap \{\sigma_g \mid \sigma \in \Sigma_0\} = \emptyset$ because otherwise $F_\Sigma \cdot^p L(A) = F_\Sigma$. We present an upper bound construction of a DBTA $B$ for $F_\Sigma \cdot^p L(A)$. Namely, $L(B) = F_\Sigma \cdot^p L(A)$. We define $B = (\Sigma, Q', Q'_F, g')$, where

$$Q' = \{X \cup \{\sigma_g \mid \sigma \in \Sigma_0\} \mid X \in 2^{Q \setminus \{\sigma_g \mid \sigma \in \Sigma_0\}}\}, Q'_F = \{q \in Q' \mid q \cap Q_F \neq \emptyset\},$$

and the transitions of $g'$ are defined as follows:

For $\tau \in \Sigma_0$, we define

$$\tau_{g'} = \{\sigma_g \mid \sigma \in \Sigma_0\}.$$

For $\tau \in \Sigma_m, m \geq 1$, and $P_1, P_2, \ldots, P_m \in Q'$, we define

$$\tau_{g'}(P_1, P_2, \ldots, P_m) = \tau_g(P_1, P_2, \ldots, P_m) \cup \{\sigma_g \mid \sigma \in \Sigma_0\}.$$

Now we explain how $B$ recognizes the tree language $F_\Sigma \cdot^p L$. Note that we define every target state of $g'$ to be the union of the set of states reachable by $g$ and the set of states reachable by reading leaf nodes. Since every target state of $g'$ is not empty, a new DBTA $B$ is complete although $A$ may not be complete. This implies that a state of $B$ contains at least the states in $\{\sigma_g \mid \sigma \in \Sigma_0\}$ that are the set of states by reading leaf nodes in $A$. After reading any tree in $F_\Sigma$, the state of $B$ contains $\{\sigma_g \mid \sigma \in \Sigma_0\}$, and thus can simulate the trees in $L(A)$. $\square$

The upper bound in Lemma 1 is reachable when a DBTA accepts a set of unary trees. If a DBTA accepts a set of unary trees, then we can regard the DBTA as a DFA with multiple initial states. Since the upper bound reaches the maximum when $k = 1$, we consider the state complexity of catenation of $L$ and $\Sigma^*$. Let $L$ be a regular language whose state complexity is $n$. Then, the state complexity of $\Sigma^* L$ is $2^{n-1}$ [19] which is the same as the bound in Lemma 1. Furthermore, we show that the upper bound is tight for any $1 \leq k \leq n$.

Choose $\Sigma = \Sigma_0 \cup \Sigma_1$, where $\Sigma_0 = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$ and $\Sigma_1 = \{a, b\}$. We define a DBTA $C_1 = (\Sigma, Q_{C_1}, Q_{C_1,F}, g_{C_1})$, where $Q_{C_1} = \{0, 1, \ldots, n-1\}, Q_{C_1,F} = \{n-1\}$ and the transition function $g_{C_1}$ is defined by setting:

- $(\sigma_i)_{g_{C_1}} = i - 1$ $(1 \leq i \leq k)$,
- $a_{g_{C_1}}(i) = i + 1 \mod n$,
- $b_{g_{C_1}}(i) = i$ $(0 \leq i < k)$,
- $b_{g_{C_1}}(i) = i + 1 \mod n$ $(k \leq i < n)$.

Based on the construction of the proof of Lemma 1, we construct a DBTA $D_1 = (\Sigma, Q_{D_1}, Q_{D_1,F}, g_{D_1})$ recognizing $F_\Sigma \cdot^p L(C_1)$, where $Q_{D_1} = \{P \mid \{0, 1, \ldots, k-1\} \subseteq P, P \subseteq Q_{C_1}\}$, $Q_{D_1,F} = \{P \mid P \in Q_{D_1}, P \cap Q_{C_1,F} \neq \emptyset\}$, and the transition function $g_{D_1}$ is defined as follows:

- $(\sigma_i)_{g_{D_1}} = \{0, 1, \ldots, k-1\}$ $(0 \leq i \leq k)$,
- $a_{g_{D_1}}(P) = a_{g_{C_1}}(P) \cup \{0, 1, \ldots, k-1\}$,
- $b_{g_{D_1}}(P) = b_{g_{C_1}}(P) \cup \{0, 1, \ldots, k-1\}$.

Notice that $L(D_1) = F_\Sigma \cdot^p L(C_1)$ by Lemma 1. In the following lemma, we establish that $D_1$ is a minimal DBTA by showing that all states of $D_1$ are reachable and pairwise inequivalent.

**Lemma 2.** *All states of $D_1$ are reachable and pairwise inequivalent.*

*Proof.* First, we prove the reachability of all states of $D_1$. Note that each state of $D_1$ is a set of states in $C_1$. By the construction, the size of a state $P$ in $Q_{D_1}$ satisfies $k \leq |P| \leq n$ since $\{0, 1, \ldots, k-1\} \subseteq P$. Using induction on $|P|$, we show that all states of $D_1$ are reachable. For the basis, we have a state $\{0, 1, \ldots, k-1\}$ of size $k$ that is reachable by reading a leaf node. Assuming that all states $P$ are reachable for $|P| \leq x$, we will show that any state $P'$ is reachable when $|P'| = x+1$. Let $P' = \{0, 1, \ldots, k-1, q_k, q_{k+1}, \ldots, q_x\}$ be a state of size $x+1$. The state $P'$ is reachable from a state $\{0, 1, \ldots, k-1, q_{k+1} - q_k + k - 1, \ldots, q_x - q_k + k - 1\}$ by reading a sequence of unary symbols $ab^{q_k - k}$. Therefore, all states are reachable by induction.

Next we prove that all states of $D_1$ are pairwise inequivalent. Pick any two distinct states $P_1$ and $P_2$. Assume $p \in P_1 \backslash P_2$. (The other possibility is completely symmetric.) After reading a sequence of unary symbols $a^{n-p-1}$, a final state is reached from state $P_1$ whereas $P_2$ reaches a non-final state. Therefore, all states of $D_1$ are pairwise inequivalent. □

Since we have shown that there exists a corresponding lower bound for the upper bound, the bound is tight.

**Theorem 1.** *Given a DBTA $A$ with $n$ states for a regular tree language $L$, $2^{n-k}$ states are necessary and sufficient in the worst-case for the minimal DBTA of $F_\Sigma \cdot^p L$ if $|\{\sigma_g \mid \sigma \in \Sigma_0\}| = k$.*

Now we consider $L \cdot^s F_\Sigma$—a tree language consists of all trees that have trees in $L$ as subtrees. In other words, for any tree $t$ in $L$, we have all possible supertrees of $t$ in $L'$. Given a regular tree language $L$, it is known that $L \cdot^s F_\Sigma$ is also a regular tree language [17]. We study the state complexity of $L \cdot^s F_\Sigma$.

**Lemma 3.** *Given a DBTA $A = (\Sigma, Q, Q_F, g)$ with $n$ states for a regular tree language $L$, $n+1$ states are sufficient for recognizing $L \cdot^s F_\Sigma$.*

*Proof.* We construct a new DBTA $B = (\Sigma, Q', Q'_F, g')$ for $L \cdot^s F_\Sigma$, where $Q' = Q \cup \{q_{\text{new}}\}$, $Q'_F = Q_F$, and the transition function $g'$ is defined as follows:
For $\tau \in \Sigma_0$, we define

$$\tau_{g'} = \begin{cases} \tau_g & \text{if } \tau_g \text{ is defined,} \\ q_{\text{new}} & \text{otherwise.} \end{cases}$$

For $\tau \in \Sigma_m, m \geq 1$, $q_1, q_2, \ldots, q_m \in Q'$, and $q_f \in Q'_F$, we define

$$\tau_{g'}(q_1, q_2, \ldots, q_m) = \begin{cases} \tau_g(q_1, q_2, \ldots, q_m) & \text{if } \tau_g(q_1, q_2, \ldots, q_m) \text{ is defined and} \\ & \{q_1, q_2, \ldots, q_m\} \cap Q_f = \emptyset, \\ q_f & \text{if } \{q_1, q_2, \ldots, q_m\} \cap Q_f \neq \emptyset, \\ q_{\text{new}} & \text{otherwise.} \end{cases}$$

Now we explain how $B$ accepts a set of all trees that are supertrees of trees in $L$. We define the transition function $g'$ to be complete by setting the target state of the undefined transition as the new state $q_{\text{new}}$. Then, $B$ moves to $q_{\text{new}}$ by reading trees in $\overline{L}$ while moving to one of its final states by reading trees in $L$. Assume that $B$ accepts a tree in $L$ and arrives at the final state $q_f$. After then, $B$ stays in $q_f$ by reading any sequence of states including the final state $q_f$. □

We cannot reach the upper bound $n+1$ with any DFA in this case since the state complexity of $L\Sigma^*$ is $n$ which is the same as that of $L$, even for the incomplete DFAs. Thus, we show that there exists a lower bound DBTA of $n+1$ states for accepting $L \cdot^s F_\Sigma$ where the state complexity of $L$ is $n$ to prove the tightness of the upper bound.

Choose $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$, where $\Sigma_0 = \{c\}$, $\Sigma_1 = \{a\}$ and $\Sigma_2 = \{b\}$. We define a DBTA $C_2 = (\Sigma, Q_{C_2}, Q_{C_2,F}, g_{C_2})$, where $Q_{C_2} = \{0, 1, \ldots, n-1\}, Q_{C_2,F} = \{n-1\}$, and the transition function $g_{C_2}$ is defined by setting:

- $c_{g_{C_2}} = 0$,
- $a_{g_{C_2}}(i) = b_{g_{C_2}}(i, i) = i + 1 \mod n$.

All transitions of $g_{C_2}$ not listed above are undefined. Based on the construction of the proof of Lemma 3, we construct a DBTA $D_2 = (\Sigma, Q_{D_2}, Q_{D_2,F}, g_{D_2})$ recognizing $L(C_2) \cdot^s F_\Sigma$, where $Q_{D_2} = Q_{C_2} \cup \{n\}$, $Q_{D_2,F} = Q_{C_2,F}$ and the transition function $g_{D_2}$ is defined as follows:

- $c_{g_{D_2}} = 0$,
- $a_{g_{D_2}}(i) = b_{g_{D_2}}(i, i) = i + 1$ $(0 \le i \le n-2)$,
- $a_{g_{D_2}}(n-1) = b_{g_{D_2}}(n-1, i) = b_{g_{D_2}}(i, n-1) = n-1$ $(0 \le i \le n-1)$,
- $a_{g_{D_2}}(n) = b_{g_{D_2}}(i, j) = n$ $(i \ne j, i \ne n-1, j \ne n-1)$.

Notice that $L(D_2) = L(C_2) \cdot^s F_\Sigma$ by Lemma 3. In the following lemma, we establish that $D_2$ is a minimal DBTA by showing that all states in $Q_{D_2}$ are reachable and pairwise inequivalent.

**Lemma 4.** *All states of $D_2$ are reachable and pairwise inequivalent.*

From two lemmas, we establish the following theorem.

**Theorem 2.** *Given a DBTA $A$ with $n$ states for a regular tree languages $L$, $n+1$ states are necessary and sufficient in the worst-case for the minimal DBTA of $L \cdot^s F_\Sigma$.*

We lastly consider the state complexity of $F_\Sigma \cdot^p L \cdot^s F_\Sigma$. Note that the sequential catenation of trees is not associative whereas the parallel catenation of trees is associative. That means that there exist trees $t_1, t_2$ and $t_3$ such that $(t_1 \cdot^s t_2) \cdot^s t_3$ and $t_1 \cdot^s (t_2 \cdot^s t_3)$ do not coincide. This also applies to the catenation of tree languages and thus, leads to $(L_1 \cdot^s L_2) \cdot^s L_3 \ne L_1 \cdot^s (L_2 \cdot^s L_3)$ for some regular tree languages $L_1, L_2$, and $L_3$. However, we consider a special tree language $F_\Sigma$ for $L_1$ and $L_3$ that makes $(F_\Sigma \cdot^s L_2) \cdot^s F_\Sigma = F_\Sigma \cdot^s (L_2 \cdot^s F_\Sigma)$. Thus, we simply denote the language by $F_\Sigma \cdot^p L \cdot^s F_\Sigma$ instead of $(F_\Sigma \cdot^s L_2) \cdot^s F_\Sigma$ or $F_\Sigma \cdot^s (L_2 \cdot^s F_\Sigma)$.

Now we tackle the state complexity of $F_\Sigma \cdot^p L \cdot^s F_\Sigma$.

**Lemma 5.** *Given a DBTA $A = (\Sigma, Q, Q_F, g)$ with $n$ states for a regular tree language $L$, $2^{n-t-k}+1$ states are sufficient for recognizing $F_\Sigma \cdot^p L \cdot^s F_\Sigma$ if $|Q_F| = t$ and $|\{\sigma_g \mid \sigma \in \Sigma_0\}| = k$.*

*Proof.* Without loss of generality, we assume that $Q_F \cap \{\sigma_g \mid \sigma \in \Sigma_0\} = \emptyset$ because otherwise $F_\Sigma \cdot^p L(A) \cdot^s F_\Sigma = F_\Sigma$. We give an upper bound construction of DBTA $B$ that recognizes $F_\Sigma \cdot^p L(A) \cdot^s F_\Sigma$. Namely, $L(B) = F_\Sigma \cdot^p L(A) \cdot^s F_\Sigma$. We define $B = (\Sigma, Q', Q'_F, g')$, where

$$Q' = \{X \cup \{\sigma_g \mid \sigma \in \Sigma_0\} \mid X \in 2^{Q \setminus (Q_F \cup \{\sigma_g \mid \sigma \in \Sigma_0\})}\} \cup \{Q_F\}, Q'_F = \{Q_F\},$$

and the transitions of $g'$ are defined as follows:

For $\tau \in \Sigma_0$, we define $\tau_{g'} = \{\sigma_g \mid \sigma \in \Sigma_0\}$.

For $\tau \in \Sigma_m, m \ge 1$, and $P_1, P_2, \ldots, P_m \in Q'$, we define

$$\tau_{g'}(P_1, P_2, \ldots, P_m) = \begin{cases} \tau_g(P_1, P_2, \ldots, P_m) \cup \{\sigma_g \mid \sigma \in \Sigma_0\} & \text{if } \bigcup_{i=1}^{m} P_i \cap Q_F = \emptyset, \\ Q_F & \text{otherwise.} \end{cases}$$

Here we do not explain how $B$ accepts $F_\Sigma \cdot^p L \cdot^s F_\Sigma$ because the construction can be explained as a simple combination of two constructions given in Lemma 1 and Lemma 3. □

We give a lower bound example that reaches the upper bound $2^{n-t-k}+1$.

Choose $\Sigma = \Sigma_0 \cup \Sigma_1$, where $\Sigma_0 = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$ and $\Sigma_1 = \{a, b, c\}$. We define a DBTA $C_3 = (\Sigma, Q_{C_3}, Q_{C_3,F}, g_{C_3})$, where $Q_{C_3} = \{0, 1, \ldots, n-1\}$, $Q_{C_3,F} = \{n-t, n-t+1, \ldots, n-1\}$ and the transition function $g_{C_3}$ is defined by setting:

- $(\sigma_i)_{g_{C_3}} = i-1$ $(1 \leq i \leq k)$,
- $a_{g_{C_3}}(i) = i+1 \mod n$,
- $b_{g_{C_3}}(i) = i$ $(0 \leq i \leq k)$,
- $b_{g_{C_3}}(i) = i+1 \mod n$ $(k \leq i < n)$,
- $c_{g_{C_3}}(i) = i+1 \mod n$ if $i \neq n-t-1$, $c_{g_{C_3}}(n-t-1) = 0$.

Based on the construction in the proof of Lemma 5, we construct a DBTA $D_3 = (\Sigma, Q_{D_3}, Q_{D_3,F}, g_{D_3})$ recognizing $F_\Sigma \cdot^p L(C_3) \cdot^s F_\Sigma$, where $Q_{D_3} = \{P \mid \{0, 1, \ldots, k-1\} \subseteq P,\ P \subseteq Q_{C_3} \setminus Q_{C_3,F}\}$, $Q_{D_3,F} = \{Q_{C_3,F}\}$, and the transition function $g_{D_3}$ is defined as follows:

- $(\sigma_i)_{g_{D_3}} = \{0, 1, \ldots, k-1\}$,
- $a_{g_{D_3}}(P) = P' \cup \{0, 1, \ldots, k-1\}$ if $a_{g_{C_3}}(P) \cap Q_{C_3,F} = \emptyset$,
- $a_{g_{D_3}}(P) = \{Q_{C_3,F}\}$ if $a_{g_{C_3}}(P) \cap Q_{C_3,F} \neq \emptyset$,
- $b_{g_{D_3}}(P) = P' \cup \{0, 1, \ldots, k-1\}$ if $b_{g_{C_3}}(P) \cap Q_{C_3,F} = \emptyset$,
- $b_{g_{D_3}}(P) = \{Q_{C_3,F}\}$ if $b_{g_{C_3}}(P) \cap Q_{C_3,F} \neq \emptyset$,
- $c_{g_{D_3}}(P) = P' \cup \{0, 1, \ldots, k-1\}$ if $c_{g_{C_3}}(P) \cap Q_{C_3,F} = \emptyset$,
- $a_{g_{D_3}}(\{Q_{C_3,F}\}) = b_{g_{D_3}}(\{Q_{C_3,F}\}) = c_{g_{D_3}}(\{Q_{C_3,F}\}) = \{Q_{C_3,F}\}$.

Notice that $L(D_3) = F_\Sigma \cdot^p L(C_3) \cdot^s F_\Sigma$ by Lemma 5. In the following lemma, we establish that $D_3$ is a minimal DBTA by showing that all states in $Q_{D_3}$ are reachable and pairwise inequivalent.

**Lemma 6.** *All states of $D_3$ are reachable and pairwise inequivalent.*

*Proof.* We prove the reachability of all non-final states of $D_3$ using induction on the size of $P$. Note that any non-final state $P \in Q_{D_3}$ satisfies $k \leq |P| \leq m-t$ because $Q_{C_3,F} \cap P = \emptyset$ and $\{\sigma_c \mid \sigma \in \Sigma_0\} \subseteq P$ by the construction. A state $\{0, 1, \ldots, k-1\}$ of size $k$ is reachable by reading a leaf node. Assume that all states $P$ is reachable for $|P| \leq x$. Then, we show that any state $P'$ of size $x+1$ is reachable. Let $P' = \{0, 1, \ldots, k-1, q_k, q_{k+1}, \ldots, q_x\}$ be a state of size $x+1$. Then, the state $P'$ is reached from a state $\{0, 1, \ldots, k-1, q_{k+1}-q_k+k-1, \ldots, q_x-q_{k+1}+k-1\}$ after reading a sequence of unary symbols $ab^{q_k-k}$. From the induction, it is easy to verify that all states except $Q_{C_3,F}$ are reachable. Furthermore, the only final state $Q_{C_3,F}$ is reachable from a non-final state $\{0, 1, \ldots, n-t-1\}$ by reading a unary symbol $a$.

Next we prove that all states of $D_3$ are pairwise inequivalent. Pick any two distinct states $P_1$ and $P_2$. Assume $p \in P_1 \setminus P_2$. (The other possibility is symmetric.) From $P_1$, a final state is reached by reading a sequence of unary symbols $c^{n-t-1-p}a$ whereas $P_2$ does not reach the final state. Therefore, any two states in $Q_{D_3}$ are pairwise inequivalent. □

**Theorem 3.** *Given a DBTA $A = (\Sigma, Q, Q_F, g)$ with $n$ states for a regular tree language $L$, $2^{n-t-k}+1$ states are necessary and sufficient in the worst-case for the minimal DBTA of $F_\Sigma \cdot^p L \cdot^s F_\Sigma$ if $|Q_F| = t$ and $|\{\sigma_g \mid \sigma \in \Sigma_0\}| = k$.*

## 5 State Complexity of DTTAs

It is well known that every NBTA can be converted into an equivalent NTTA [1,4]. However, it does not mean that there always exists a DTTA for every regular tree language. This implies that a class of regular tree languages accepted by DTTAs is a proper subclass of regular tree languages accepted by NBTAs and NTTAs. It is known that DTTAs recognize exactly the class of *path-closed languages* which is a proper subclass of regular tree languages [1,4].

We study the state complexity of path-closed languages for tree matching. Following the previous results, we consider three types of tree languages $F_\Sigma \cdot^p L$, $L \cdot^s F_\Sigma$, and $F_\Sigma \cdot^p L \cdot^s F_\Sigma$, where $L$ is a tree language. However, two tree languages $L \cdot^s F_\Sigma$ and $F_\Sigma \cdot^p L \cdot^s F_\Sigma$ appear to be not path-closed languages. Nivat and Podelski [13] argued that path-closed languages can be characterized by a property called the subtree exchange property as follows:

**Corollary 1 (Nivat and Podelski [13]).** *A regular tree language $L$ is path-closed if and only if, for every $t \in L$ and every node $u \in t$, if $t(u \leftarrow a(t_1, \ldots, t_m)) \in L$ and $t(u \leftarrow a(s_1, \ldots, s_m)) \in L$, then $t(u \leftarrow a(t_1, \ldots, s_i, \ldots, t_m)) \in L$ for each $i = 1, \ldots, m$.*

Using the subtree exchange property, we prove that given a tree language $L$, $L \cdot^s F_\Sigma$ and $F_\Sigma \cdot^p L \cdot^s F_\Sigma$ are not path-closed languages.

**Lemma 7.** *There exists a path-closed language $L$ such that $L \cdot^s F_\Sigma$ is not a path-closed language.*

*Proof.* Let $\Sigma = \Sigma_2 \cup \Sigma_0$, where $\Sigma_2 = \{b\}$, and $\Sigma_0 = \{a, c\}$. A singleton language $L$ contains a single-node tree $c$, namely $L = \{c\}$. It is straightforward to verify that $F_\Sigma$ contains every binary tree where leaf nodes are labeled by $a$ or $c$, and non-leaf nodes are labeled by $b$. Then, $L \cdot^s F_\Sigma$ is a set of binary trees where every tree contains at least one leaf labeled by $c$. Therefore, $b(a, c) \in L \cdot^s F_\Sigma$, $b(c, a) \in L \cdot^s F_\Sigma$, and $b(a, a) \notin L \cdot^s F_\Sigma$ hold. However, if $L \cdot^s F_\Sigma$ is path-closed, $b(a, a)$ should exist in $L \cdot^s F_\Sigma$ by the subtree exchange property. This implies that $L \cdot^s F_\Sigma$ is not a path-closed language. □

**Lemma 8.** *There exists a path-closed language $L$ such that $F_\Sigma \cdot^p L \cdot^s F_\Sigma$ is not a path-closed language.*

*Proof.* Let $\Sigma = \Sigma_2 \cup \Sigma_0$, where $\Sigma_2 = \{a, b\}$, and $\Sigma_0 = \{c\}$. A singleton language $L$ contains a tree $a(c, c)$, namely $L = \{a(c, c)\}$. It is easy to verify that $F_\Sigma$ contains every binary tree where all leaf nodes are labeled by $c$ and non-leaf nodes are labeled by $a$ or $b$.

Then, $F_\Sigma \cdot^p L \cdot^s F_\Sigma$ is a set of binary trees where every tree contains at least one non-leaf node labeled by $a$. Therefore, $b(a(c, c), c) \in F_\Sigma \cdot^p L \cdot^s F_\Sigma, b(c, a(c, c)) \in F_\Sigma \cdot^p L \cdot^s F_\Sigma$, and $b(c, c) \notin F_\Sigma \cdot^p L \cdot^s F_\Sigma$. However, due to the subtree exchange property, $b(c, c)$ should be in $F_\Sigma \cdot^p L \cdot^s F_\Sigma$ if the language $F_\Sigma \cdot^p L \cdot^s F_\Sigma$ is path-closed. This means that $F_\Sigma \cdot^p L \cdot^s F_\Sigma$ is not a path-closed language. □

We define the *deterministic top-down state complexity* of a path-closed language $L$ to be the number of states that are necessary and sufficient in the worst-case for the minimal DTTA recognizing $L$.

**Theorem 4.** *Given a DTTA $A = (\Sigma, Q, Q_0, g)$ with $n$ states for a path-closed language $L$, $n$ states are necessary and sufficient in the worst-case for the minimal DTTA of $F_\Sigma \cdot^p L$.*

*Proof.* We construct a new DTTA $B = (\Sigma, Q', Q'_0, g')$ for $F_\Sigma \cdot^p L$, where $Q' = Q$, $Q'_0 = Q_0$, and the transition function $g'$ is defined as follows:

For $\tau \in \Sigma_m, m \geq 0$ and $q \in Q'$, we define

$$\tau_{g'}(q) = \begin{cases} \tau_g(q) & \text{if } \sigma_g(q) \neq \lambda \text{ for any } \sigma \in \Sigma_0, \\ \underbrace{[q, q, \ldots, q]}_{m \text{ times}} & \text{otherwise.} \end{cases}$$

Now we explain how $B$ simulates $F_\Sigma \cdot^p L$ with $n$ states. Since trees in $F_\Sigma \cdot^p L$ have the same topmost parts with trees in $L$ and leaves can be substituted with any tree in $F_\Sigma$, $B$ simulates from the same initial state with $A$. Let us assume that a state $q \in Q'$ may end the top-down computation with generating a leaf node since $\sigma_g(q) = \lambda$. Once $B$ arrives at $q$, the new transition function $g'$ continues the computation by reading a non-leaf label of rank $m$ and generating a sequence $[q, q, \ldots, q]$ of states whose length is $m$. This makes a new DTTA $B$ to generate any subtree in $F_\Sigma$ at the point where the computation may end with generating leaves and, thus, recognize the language $F_\Sigma \cdot^p L$.

It is easy to verify that $n$ states are necessary to recognize $F_\Sigma \cdot^p L$. Consider a path-closed language of unary trees whose state complexity correspond to that of regular string languages. Since the state complexity of $L\Sigma^*$ is $n$ if the state complexity of $L$ is $n$, this case can be a lower bound for the path-closed language $F_\Sigma \cdot^p L$. $\square$

## 6 Conclusions

We have considered three tree languages $F_\Sigma \cdot^p L$, $L \cdot^s F_\Sigma$, and $F_\Sigma \cdot^p L \cdot^s F_\Sigma$ motivated from the tree pattern matching problem and have established the state complexity of these languages described by DBTAs and DTTAs. We have also shown that $L \cdot^s F_\Sigma$ and $F_\Sigma \cdot^p L \cdot^s F_\Sigma$ are not recognizable by DTTAs even when $L$ is a path-closed language since they are not necessarily path-closed languages. In addition, we have demonstrated that $L \cdot^s F_\Sigma$ and $F_\Sigma \cdot^p L \cdot^s F_\Sigma$ need not be path-closed and therefore cannot recognized by DTTAs. In future, we aim to investigate the descriptional complexity of unranked tree automata, which are a more generalized model than tree automata over ranked alphabet, for recognizing $L \cdot^s F_\Sigma$ and $F_\Sigma \cdot^p L \cdot^s F_\Sigma$.

## References

1. Comon, H., Dauchet, M., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M.: Tree Automata Techniques and Applications (2007), Electronic book available at `http://www.tata.gforge.inria.fr`

2. Crochemore, M., Hancart, C.: Automata for Matching Patterns. In: Handbook of formal languages, vol. 2, pp. 399–462 (1997)
3. Eom, H.-S., Han, Y.-S., Ko, S.-K.: State complexity of subtree-free regular tree languages. In: Jurgensen, H., Reis, R. (eds.) DCFS 2013. LNCS, vol. 8031, pp. 66–77. Springer, Heidelberg (2013)
4. Gécseg, F., Steinby, M.: Tree languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages. Beyond Words, vol. 3, pp. 1–68. Springer-Verlag New York, Inc., (1997)
5. Hoffmann, C.M., O'Donnell, M.J.: Pattern matching in trees. Journal of the ACM 29(1), 68–95 (1982)
6. Holzer, M., Kutrib, M.: Descriptional and computational complexity of finite automata – a survey. Information and Computation 209, 456–470 (2011)
7. Kutrib, M., Pighizzini, G.: Recent trends in descriptional complexity of formal languages. Bulletin of the EATCS 111, 70–86 (2013)
8. Lupanov, O.: A comparison of two types of finite sources. Problemy Kibernetiki 9, 328–335 (1963)
9. Maslov, A.: Estimates of the number of states of finite automata. Soviet Mathematics Doklady 11, 1373–1375 (1970)
10. Meyer, A., Fisher, M.: Economy of description by automata, grammars and formal systems. In: Proceedings of the 12th Annual Symposium on Switching and Automata Theory, pp. 188–191 (1971)
11. Moore, F.: On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic and two-way finite automata. IEEE Transactions on Computers C-20, 1211–1214 (1971)
12. Neven, F.: Automata theory for XML researchers. ACM SIGMOD Record 31(3), 39–46 (2002)
13. Nivat, M., Podelski, A.: Minimal ascending and descending tree automata. SIAM Journal on Computing 26(1), 39–58 (1997)
14. Piao, X., Salomaa, K.: State trade-offs in unranked tree automata. In: Holzer, M., Pighizzini, G., kutrib, M. (eds.) DCFS 2011. LNCS, vol. 6808, pp. 261–274. Springer, Heidelberg (2011)
15. Piao, X., Salomaa, K.: Transformations between different models of unranked bottom-up tree automata. Fundamenta Informaticae 109(4), 405–424 (2011)
16. Piao, X., Salomaa, K.: State complexity of Kleene-star operations on trees. In: Dinneen, M.J., Khoussainov, B., Nies, A. (eds.) WTCS 2012 (Calude Festschrift). LNCS, vol. 7160, pp. 388–402. Springer, Heidelberg (2012)
17. Piao, X., Salomaa, K.: State complexity of the concatenation of regular tree languages. Theoretical Computer Science 429, 273–281 (2012)
18. Shallit, J.: A Second Course in Formal Languages and Automata Theory, 1st edn. Cambridge University Press, New York (2008)
19. Yu, S., Zhuang, Q., Salomaa, K.: The state complexities of some basic operations on regular languages. Theoretical Computer Science 125(2), 315–328 (1994)